VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Dominik Indrák

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV RADIOELEKTRONIKY
DEPARTMENT OF RADIOENGINEERING

REKONFIGUROVATELNÝ GENERÁTOR 5G NR SIGNÁLŮ NA RFSOC FPGA
RECONFIGURABLE 5G NR SIGNAL GENERATOR ON RFSOC FPGA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Dominik Indrák**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. Roman Maršálek, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Elektronika a sdělovací technika**

Ústav radioelektroniky

Student: Bc. Dominik Indrák ***ID:*** 186097
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Rekonfigurovatelný generátor 5G NR signálů na RFSoC FPGA

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte materiály popisující připravovaný standard 5G NR (LTE Rel. 15/16) s důrazem na fyzickou vrstvu (PHY). Simulujte základní strukturu modulátoru a demodulátoru OFDM pro 5G NR v prostředí MATLAB. Seznamte se s obvody Xilinx SoC a RFSoC, s jejich vlastnostmi, vývojovým prostředím a jeho konfigurací. Navrhněte blokovou strukturu výsledného generátoru, umožňujícího generovat RF signály OFDM dle standardu 5G NR. Navrhněte metodiku pro ověření činnosti generátoru jednoduchým přijímačem implementovaným např. v GNU radiu, prostředí MATLAB nebo s využitím bloků Xilinx System Generatoru.

S využitím platformy Xilinx RFSoC, v případě její nedostupnosti s využitím Xilinx SoC s D/A převodníky, implementujte zjednodušený generátor 5G NR OFDM signálu. Ve zvoleném prostředí (např. GNU radio, MATLAB nebo System Generator) vytvořte softwarový demodulátor OFDM signálu, pomocí něhož ověřte činnost vámi navrženého generátoru. Vypracujte podrobnou technickou zprávu.

DOPORUČENÁ LITERATURA:

[1] ETSI TS 138 211 v 15.2.0 - 5G NR Physical channels and modulation. Sophia Antipolis Cedex: ETSI. 2018

[2] ZCU111 Evaluation Board, User Guide, Xilinx, 2018, [online]. Dostupné z https://www.xilinx.com/support/documentation/boards_and_kits/zcu111/ug1271-zcu111-eval-bd.pdf

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 28.5.2020

Vedoucí práce: prof. Ing. Roman Maršálek, Ph.D.

prof. Ing. Tomáš Kratochvíl, Ph.D.
předseda oborové rady

ABSTRAKT

Práce se zabývá simulací základní struktury OFDM modulátoru a demodulátoru připravovaného standardu 5G NR. V prostředí MATLAB jsou simulovány základní bloky jako je modulace, vkládání referenčních signálů, Fourierova transformace, vkládání cyklického prefixu, AWGN a vícecestné šíření. V práci je navržen způsob implementace modulátoru a demodulátoru do RFSoC kitu a jeho konfigurace. S využitím platformy STEMLab RedPitaya je implementován navržený generátor. V programu Matlab je generován 5G OFDM signál určený k vysílání. Přijatý signál je pak vyhodnocován opět v programu Matlab.

KLÍČOVÁ SLOVA

5G, 5G NR, OFDM, RFSoC, RFDC, DUC, DDC, FPGA, Rekonfigurovatelný generátor, RedPitaya, STEMLab RedPitaya, Vivado, IP Integrator

ABSTRACT

This work deal with simulation of basic structure of OFDM modulator and demodulator of the upcoming standard 5G NR. In MATLAB are simulated basic parts including modulation, reference signal inserting, Fourier transform, cyclic prefix inserting, AWGN and multi-path propagation. In this work is proposed implementation of the modulator and demodulator into RFSoC board and his configuration. Designed generator is implemented with the use of STEMLab RedPitaya platform. In Matlab software is generated 5G OFDM signal used to transmitt. Received signal is evaluated in Matlab software.

KEYWORDS

5G, 5G NR, OFDM, RFSoC, RFDC, DUC, DAC, FPGA, Reconfigurable generator, RedPitaya, STEMLab RedPitaya, Vivado, IP Integrator

INDRÁK, Dominik. *Rekonfigurovatelný generátor 5G NR signálů na RFSoC FPGA*. Brno, 2020, 95 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky. Vedoucí práce: prof. Ing. Roman Maršálek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Rekonfigurovatelný generátor 5G NR signálů na RFSoC FPGA“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Tímto bych rád poděkoval vedoucímu diplomové práce panu prof. Ing. Romanovi Maršálkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k diplomové práci. Dále bych rád poděkoval panu Ing. Miroslavu Waldeckerovi za velmi cenné rady týkající se implementace v RedPitaya.

Brno ....

podpis autora(-ky)

OBSAH

Úvod **12**

1 Teorie **13**
1.1 Standard 5G – Fyzická vrstva . . . 13
1.1.1 Obecný popis a numerologie . . . 13
1.1.2 Struktura rámce . . . 14
1.1.3 Resource Grid . . . 16
1.1.4 Resource Block . . . 16
1.1.5 Bandwidth part . . . 17
1.1.6 Referenční signály . . . 18
1.1.7 Modulace . . . 21
1.1.8 OFDM . . . 22
1.1.9 Vyrovnávače kanálu . . . 23
1.2 ZYNQ Ultra Scale . . . 26
1.2.1 RF Data Converter . . . 27
1.2.2 RFDC tok dat . . . 29
1.2.3 Hodinové signály a časování . . . 31
1.2.4 Možné způsoby konfigurace RFDC . . . 33
1.3 STEMLab Red Pitaya a Zynq XC7Z010 . . . 39
1.3.1 Processing system (PS) . . . 40
1.3.2 Programable Logic (PL) . . . 41

2 Řešení **43**
2.1 Simulace v prostředí MATLAB . . . 43
2.2 Koncepce generátoru 5G signálů . . . 51
2.2.1 Převod signálu ze základního pásma na nosný kmitočet . . . 52
2.2.2 Konfigurace RFDC . . . 54
2.3 Implementace generátoru . . . 55
2.3.1 Matlab . . . 56
2.3.2 C server . . . 58
2.3.3 Implementace ve Vivadu . . . 60
2.4 Činnost generátoru a výsledky . . . 75
2.4.1 RedPitaya - připojení a nahrání bitstreamu . . . 75
2.4.2 Generování a příjem dat . . . 77

3 Závěr **84**

Literatura **86**

Seznam symbolů, veličin a zkratek **90**

Seznam příloh **92**

A Tabulky ke konfiguraci RFDC **93**

B Blokové schéma implementace **95**

SEZNAM OBRÁZKŮ

1.1 Struktura rámce [2] . . . 15
1.2 Struktura rámce s cyklickým prefixem pro $\Delta f = 15$ kHz [3] . . . 15
1.3 Resource Grid [1, ResourceGrid] . . . 17
1.4 Příklad využití šířky pásma kanálu [6, BWP] . . . 18
1.5 Mapovací typ A [1, PDSCH DMRS] . . . 19
1.6 Mapovací typ B [1, PDSCH DMRS] . . . 20
1.7 Typ 1 a Typ 2 [1, PDSCH DMRS] . . . 20
1.8 Konstelační diagramy modulací (BPSK, QPSK, 16QAM, 64QAM) . 21
1.9 OFDM modulátor . . . 22
1.10 OFDM demodulátor . . . 23
1.11 Přenosová a vyrovnávací funkce kanálu . . . 24
1.12 MMSE vyrovnávač [12, Vyrovnavače kanálu] . . . 24
1.13 Přehled parametrů čipů dostupných na Zynq UltaScale+ RFSoC [8] 26
1.14 RF Data Converter [10, RFDC LogiCORE IP] . . . 27
1.15 Struktura ADC+DDC [10, RFDC LogiCORE IP] . . . 28
1.16 Struktura DAC+DUC [10, RFDC LogiCORE IP] . . . 28
1.17 Tok dat u DAC . . . 29
1.18 Tok dat u ADC . . . 30
1.19 Zdroje hodinového signálu ZCU111 [10, RFDC LogiCORE IP] . . . 31
1.20 Struktura hodinových domén v dlaždici [10, RFDC LogiCORE IP] . 32
1.21 Struktura PLL [10, RFDC LogiCORE IP] . . . 32
1.22 Okno Radio Frequency Data Converter (RFDC) IP jádra pro nastavení ADC [10, RFDC LogiCORE IP] . . . 34
1.23 Okno RFDC IP jádra pro nastavení DAC [10, RFDC LogiCORE IP] 36
1.24 Okno RFDC IP Coru pro nastavení hodin [10, RFDC LogiCORE IP] 37
1.25 Okno RFDC IP Coru pro pokročilé nastavení [10, RFDC LogiCORE IP] . . . 38
1.26 Blokové schéma vývojového kitu RedPitaya [16] . . . 39
1.27 Vnitřní struktura čipu XC7Z010 [13] . . . 40
1.28 Struktura MIO [13] . . . 41
2.1 Blokové znázornění toku dat při simulaci v prostředí Matlab . . . 43
2.2 I/Q diagram 16QAM modulovaných dat . . . 44
2.3 Použitá zjednodušená konfigurace DMRS . . . 45
2.4 Příklad spektrální hustoty výkonu OFDM signálu . . . 45
2.5 Závislost BER na SNR pro OFDM s 4096 subnosnými, dva modely kanálu, modulace 16QAM, bez kódování . . . 47
2.6 Spektrální hustota výkonu signálu po průchodu kanálem . . . 48

2.7 I/Q diagram signálu po průchodu kanálem 48
2.8 Modulová část přenosové funkce kanálu po průchodu kanálem a vyrovnávací funkce 49
2.9 I/Q diagram opravených dat 50
2.10 Blokové schéma koncepce generátoru, první varianta 51
2.11 Blokové schéma koncepce generátoru, druhá varianta 52
2.12 První typ up-konverze 52
2.13 Druhý typ up-konverze 53
2.14 Třetí typ up-konverze 53
2.15 Operace DAC v Nyquistových zónách [10, RFDC LogiCORE IP] . . . 54
2.16 Blokové schéma výsledného generátoru 55
2.17 Způsob prokládání dat 57
2.18 Editor adres ve Vivadu 58
2.19 Konfigurace FIFO paměti 62
2.20 Konfigurace Rozdělovače datového toku 63
2.21 Modulová charakteristika navržené dolní propusti pro FIR filtr . . . 63
2.22 Konfigurace FIR kompilátoru 65
2.23 Konfigurace dat FIR kompilátoru 65
2.24 Konfigurace DDS 66
2.25 Konfigurace DDS - výstupní frekvence 67
2.26 Zapojení DDS a bloků pro výběr bitů 67
2.27 Konfigurace bloku pro výběr bitů pro signál sinus a cosinus 68
2.28 Konfigurace Násobičky 68
2.29 Konfigurace Sčítačky 69
2.30 Konfigurace Převodníku podmnožin 70
2.31 Konfigurace Převodníku podmnožin 71
2.32 Konfigurace FIR kompilátoru pro decimaci 72
2.33 Konfigurace dat FIR kompilátoru pro decimaci 72
2.34 Konfigurace Slučovače datového toku 73
2.35 Konfigurace Převodníku bitové šířky 73
2.36 Konfigurace Putty pro připojení k Red Pitaya 75
2.37 I/Q diagram 16QAM modulovaných dat 78
2.38 Spektrální hustota výkonu vysílaných dat 78
2.39 Odhad spektra dat na výstupu DAC změřený real–time spektrálním analyzátorem FSVR 79
2.40 Reálná a imaginární složka dat před časovou synchronizací 80
2.41 Reálná a imaginární složka dat po časové synchronizaci 80
2.42 Spektrální hustota výkonu přijatého signálu 81

2.43 I/Q diagram 16QAM modulovaných dat po průchodu přenosovým kanálem bez úpravy vyrovnávačem 82
2.44 Modulová část přenosové funkce kanálu 82
2.45 Modulová část vyrovnávací funkce kanálu 83
2.46 I/Q diagram opravených 16QAM modulovaných dat 83
B.1 Blokové schéma implementace v programu Vivado 95

SEZNAM TABULEK

1.1 Podporovaná numerologie [2] 14
1.2 Délky cyklických prefixů 16
A.1 Zvolená konfigurace pro Analog to Digital Converter (ADC) 93
A.2 Zvolená konfigurace pro Digital to Analog Converter (DAC) 94

ÚVOD

První generace mobilních technologií se zaměřovaly na přenos hlasu. Zlom nastal generací 2G, kdy byla možnost posílat textové zprávy, ve 3G pak video-hovory. Ve 4G je k dispozici připojení, které umožňuje být stále online. S vývojem stále nových technologií je požadavek přenášet veliké objemy dat. Rychlý přenos velkého objemu dat je pak využíván pro služby jako je např. sdílení dat. Nedílná součást je také vysoká spolehlivost připojení a jeho velmi malá latence. Právě vznikající telekomunikační standard 5G se proto zaměřuje na:

- Vysokorychlostní datové služby, které souvisí se stále rostoucí poptávkou po rychlejším a objemnějším datovém přístupu.
- Masivní IoT, které počítá s velkým množstvím zařízení a strojů komunikujících mezi sebou (oblast automatizace).
- Velmi spolehlivé služby s nízkou latencí, což znamená výměnu dat u kritických komunikací jako je například monitorování zdraví.

1 TEORIE

1.1 Standard 5G – Fyzická vrstva

1.1.1 Obecný popis a numerologie

Napříč celým standardem je používána časová jednotka, která je definována jako [2]

$$T_c = \frac{1}{\Delta f_{max} \cdot N_f} \tag{1.1}$$

kde $\Delta f_{max} = 480$ kHz, $N_f = 4096$. Pak tedy $T_c = 0.509$ ns. Je to vlastně nejmenší možná vzorkovací perioda, která se získá po dosazení maximální vzdálenosti nosných a maximální velikosti FFT. Na základě toho můžeme určit vztah mezi 5G a LTE, jestliže víme, že pro LTE je vzorkovací perioda definována jako [2]

$$T_s = \frac{1}{\Delta f_{ref} \cdot N_{f,ref}} \tag{1.2}$$

kde $\Delta f_{ref} = 15$ kHz, $N_{f,ref} = 2048$. Pak tedy $T_s = 32.522$ ns. Nejnižší vzorkovací perioda u 5G je pak 64x kratší oproti vzorkovací periodě LTE, což je poměr mezi T_s a T_c.

Jedním z rozdílů mezi 5G a LTE je, že 5G využívá více variant vzdáleností nosných, zatímco LTE má pouze jednu. Z toho tedy plynou odlišnosti jak v době trvání OFDM symbolu, tak i v době trvání cyklického prefixu. Použitím různé numerologie tak lze dosáhnout větší flexibility spojení, která je třeba pro dodržení protichůdných požadavků v 5G. Je to např. dosažení požadovaného QoS, velmi nízké latence a vysoké spolehlivosti popřípadě vysoké přenosové rychlosti a kapacity spojení. Použitím menší vzdálenosti nosných lze dosáhnout delšího OFDM symbolu, což je užitečné pro potlačení InterSymbol Interference (ISI). To ale na druhou stranu vede na větší šířku pásma, delší latenci a snižuje propustnost celého systému. Naopak větší vzdáleností nosných lze dosáhnout delších cyklických prefixů a je tím pádem možné lépe potlačit vícecestné šíření. Splnění protichůdných požadavků je dosaženo tím, že se do šířky pásma kanálu umístí více numerologií. Tím se zabývá kapitola 1.1.3. Podporované numerologie jsou uvedeny v tab. 1.1, kde μ udává konfiguraci pro vzdálenost mezi nosnými.

μ	$\Delta f = 2^{\mu} \cdot 15$ [kHz]	Cyklický prefix
0	15	Normální
1	30	Normální
2	60	Normální, rozšířený
3	120	Normální
4	240	Normální

Tab. 1.1: Podporovaná numerologie [2]

1.1.2 Struktura rámce

Základní časovou jednotkou je rámec, který má dobu trvání 10 ms. Každý rámec je dále dělen na polovinu (tedy na 2x5 ms). Každý jeden rámec obsahuje 10 subrámců a doba trvání každého subrámce je 1 ms. Od subrámce už je struktura oproti jiným standardům flexibilní. Subrámce jsou tvořeny sloty a počet slotů v subrámci není konstantní. Je závislý na zvolené numerologii. Naopak 1 slot obsahuje vždy 14 OFDM symbolů pro normální cyklický prefix a 12 OFDM symbolů pro rozšířený cyklický prefix. Popsanou strukturu lze vidět na obr. 1.1. Cyclic Prefix (CP) je pak vkládán mezi jednotlivé OFDM symboly.

Hodnotu CP můžeme určit z rovnice

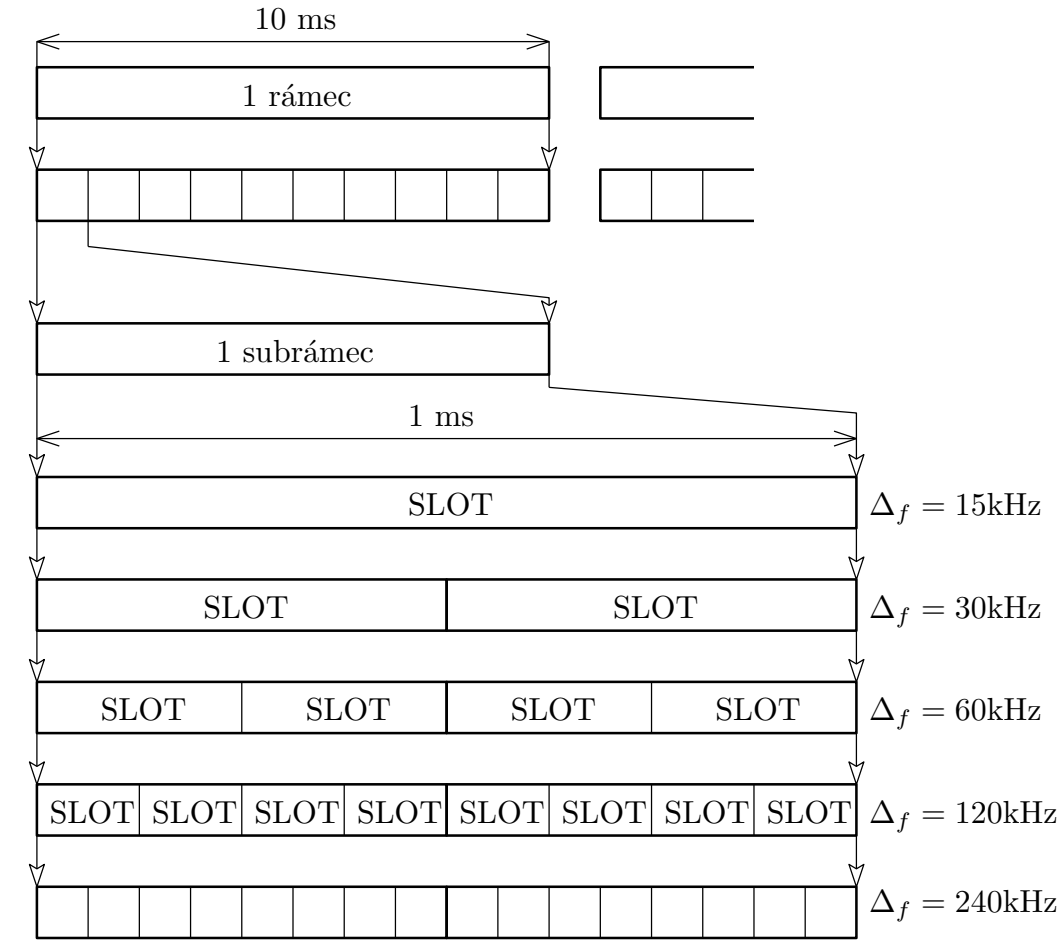
$$T_{subframe} = T_{vz} \cdot \left(N_{symb}^{subframe} \cdot N_{fft} + N_{cp} \left(N_{symb}^{subframe} - 2 \right) + 2 \cdot N_{cp\_1st} \right), \tag{1.3}$$

kde N_{cp_1st} je délka prvního CP a N_{cp} je délka všech dalších CP v jedné polovině subrámce. Pro výpočet je nutná vzorkovací perioda a počet symbolů na subrámec.

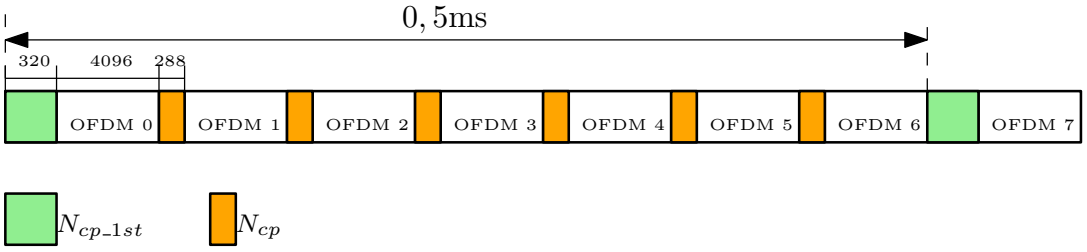
$$T_{vz} = \frac{1}{\Delta f \cdot N_{fft}} \tag{1.4}$$

$$N_{symb}^{subframe} = 2^{\mu} \cdot N_{symb}^{slot} \tag{1.5}$$

Grafické znázornění rozložení OFDM symbolů a CP pro $\Delta f = 15$ kHz a $N_{fft} = 4096$ lze vidět na obr. 1.2. Hodnoty délky CP pro kombinace Δf a N_{fft} jsou v tab. 1.2



Obr. 1.1: Struktura rámce [2]



Obr. 1.2: Struktura rámce s cyklickým prefixem pro $\Delta f = 15$ kHz [3]

	N_{FFT}	Δf [kHz]				
		15	30	60	120	240
N_{cp_1st}	4096	320	352	416	544	800
N_{cp}		288				

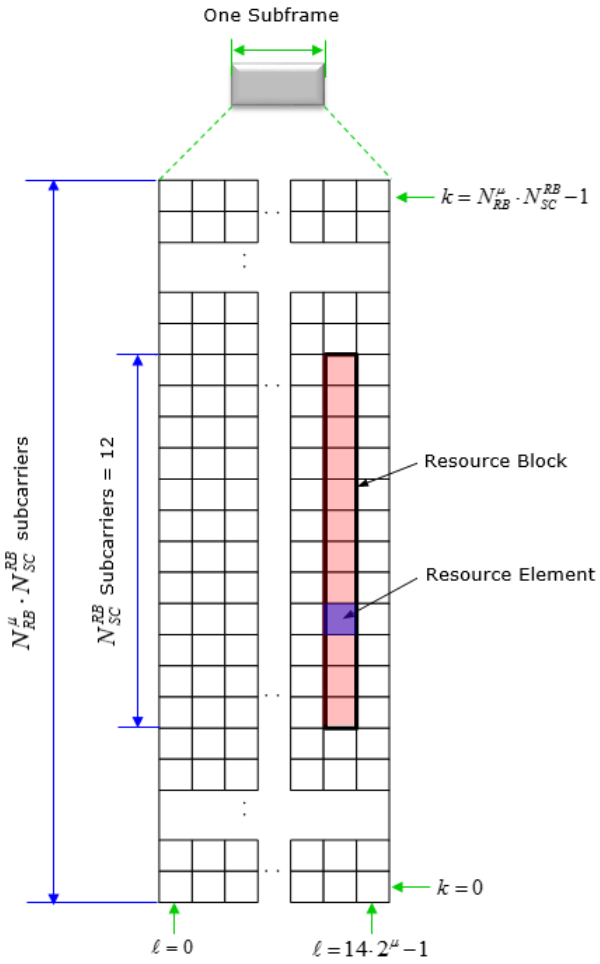
Tab. 1.2: Délky cyklických prefixů

1.1.3 Resource Grid

Jak je popsáno výše, šířka pásma kanálu může být rozdělena na více částí, kde každá část může mít jinou numerologii. Taková část je definována jako Resource Grid (RG), kterou lze vidět na obr. 1.3. RG je pomyslná mřížka, kde její sloupce jsou definovány v časové oblasti a řádky ve frekvenční oblasti. Sloupce jsou tedy tvořeny OFDM symboly ze subrámce. Počet symbolů v subrámci je závislý na zvolené numerologii, proto počet sloupců nebude konstantní. Řádky RG jsou tvořeny subnosnými, které se sdružují do bloků s názvem Resource Block (RB). Je definována sada RG jak pro DL tak pro UL, ale pro daný anténní port je pouze jeden RG. V rámci RG je definován Resource Element (RE), což je vlastně jedna dvourozměrná hodnota z RG, která je určena číslem OFDM symbolu a číslem subnosné.

1.1.4 Resource Block

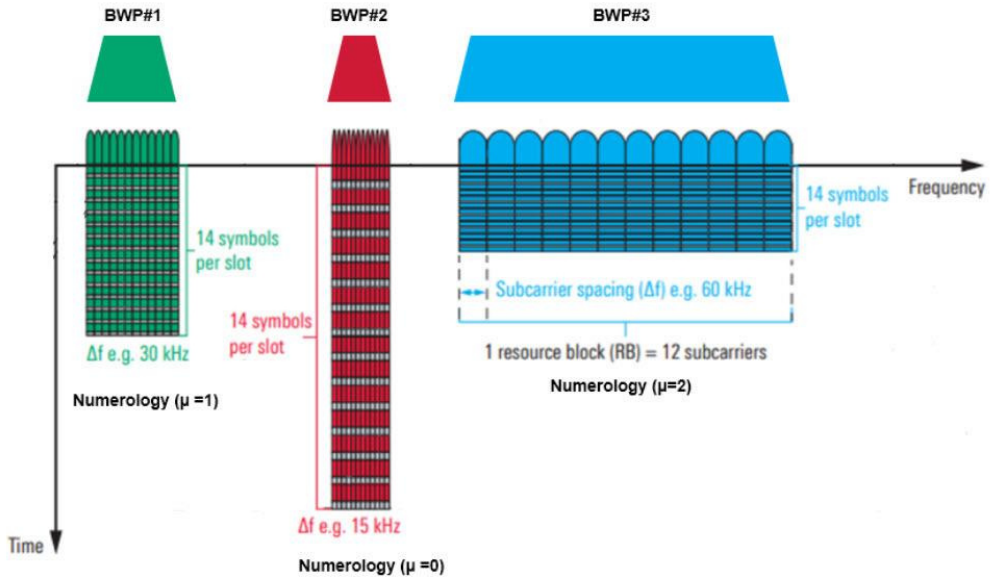
RB je definován jako 12 sousedících subnosných ve frekvenční doméně. Dále je definován referenční bod A. Je to bod, který je umístěn ve středu nulté subnosné nejnižšího RG. Od tohoto bodu začíná i šířka pásma kanálu a tento referenční bod je signalizován parametrem z vyšších vrstev komunikačního modelu. Pro číslování RB a tedy i RG napříč celou šířkou pásma slouží Common Resource Block (CRB). Ten začíná na indexu nula a končí na indexu, který je dán počtem RB v šířce pásma. Může být chápán jako orientační ukazatel ve frekvenční doméně. S CRB souvisí Physical Resource Block (PRB). PRB čísluje RB v rámci RG, je závislý na numerologii a je tedy relativní v počátku šířky pásma kanálu. Pomocí CRB a PRB se tak můžeme pohybovat v rámci celé šířky pásma i přes různé numerologie.



Obr. 1.3: Resource Grid [1, ResourceGrid]

1.1.5 Bandwidth part

Část využité šířky pásma, která používá stejnou numerologii se nazývá BandWidth Part (BWP). V jeden okamžik je aktivní pouze jedna pro UL nebo DL. BWP musí být definována v rámci RG a musí mít i stejnou numerologií jako RG. BWP pak používá PRB jako index. CRB tedy čísluje od referenčního bodu A přes celou šířku pásma a PRB čísluje od začátku do konce dané BWP. PRB a CRB jsou vůči sobě frekvenčně zarovnané. BWP nemůže zabírat dva RG, protože by tím byl porušen požadavek, aby byla aktivní pouze jedna numerologie. BWP může zabírat například třetinu celkové šířky pásma. Ve druhé třetině pak může probíhat komunikace s jinou numerologií a ve třetí třetině komunikace na jiném principu. BWP jsou vidět na obr. 1.4.



Obr. 1.4: Příklad využití šířky pásma kanálu [6, BWP]

1.1.6 Referenční signály

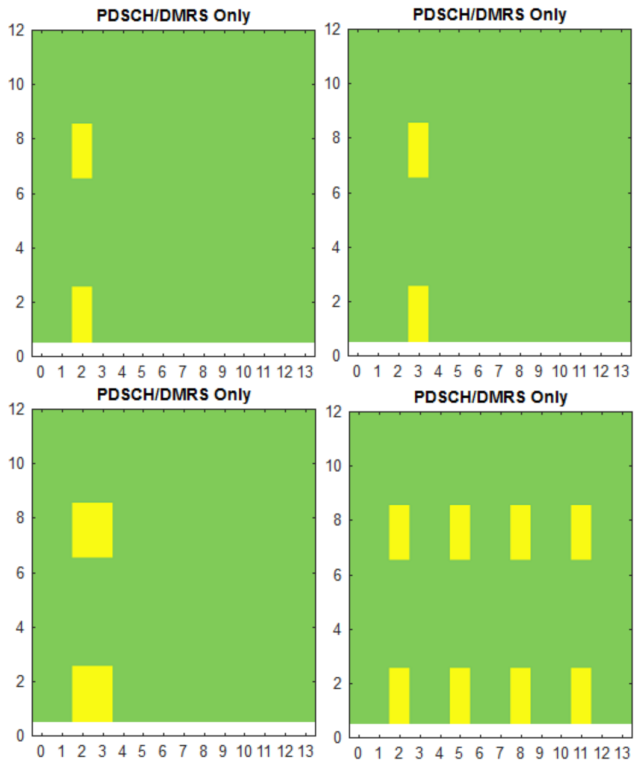
DMRS

Je určen pro odhad rádiového kanálu při demodulaci. DeModulation Reference Signal (DMRS) je specifický pro konkrétní User Equipment (UE) a může být vysílán pouze v případě potřeby (oproti Long Term Evolution (LTE)) a to buď v UL nebo v DL. Je možné alokovat více ortogonálních DMRS pro podporu Multiple Input Multiple Output (MIMO). DMRS definujeme pro časovou doménu (vodorovná osa v obr. 1.5 až obr. 1.7) a pro frekvenční doménu (svislá osa v obr. 1.5 až obr. 1.7). Pro definici v časové doméně existuje Mapovací Typ A a Mapovací Typ B (specifikují hustotu DMRS v časové doméně). Pro definici ve frekvenční doméně existuje Typ 1 a Typ 2 (specifikují hustotu DMRS ve frekvenční doméně a ovlivňují počet možných ortogonálních sekvencí). Jejich vlastnosti jsou:

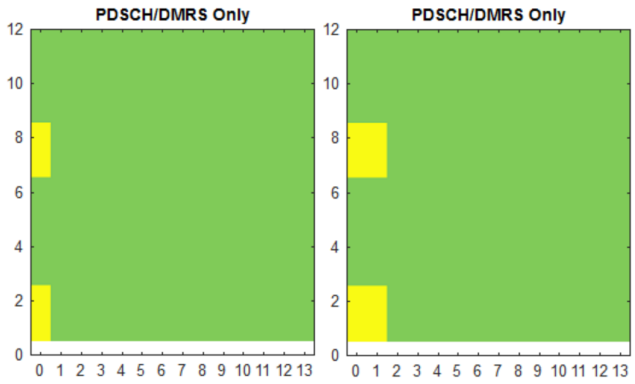
- Mapovací Typ A – DMRS symbol může začínat pouze na druhém (první obrázek na obr. 1.5) nebo třetím (druhý obrázek na obr. 1.5) symbolu slotu. Tento typ nemůže být tedy použit, pokud Physical Downlink Shared Channel (PDSCH) alokace začíná na symbolu >3 slotu. Další stupeň volnosti je, zda alokovat jednonásobné nebo dvojnásobné DMRS symboly (1 symbol nebo 2 sousedící symboly). To lze vidět na třetím obrázku na obr. 1.5. Dále se může měnit počet DMRS symbolů ve slotu (až 3 přídavné DMRS symboly), tzn. až

čtyři DMRS symboly ve slotu (čtvrtý obrázek na obr. 1.5). Větší počet DMRS symbolů pomáhá s odhadem rádiového kanálu při vysokých rychlostech.

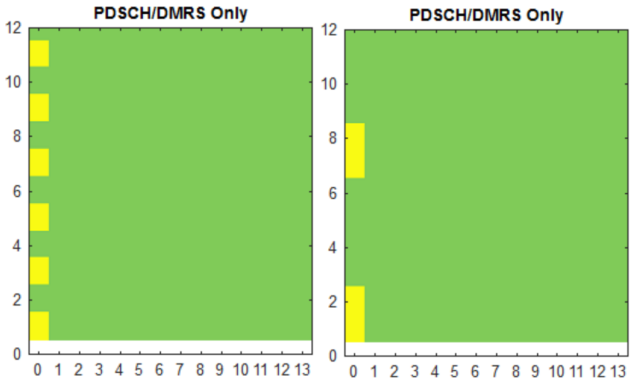
- Mapovací Typ B – DMRS symbol může začínat pouze na prvním symbolu PDSCH alokace. Počet přídavných symbolů ve slotu závisí u Typu A i B na počtu alokovaných OFDM symbolů ve slotu. Je možné použít buď jednonásobný nebo dvojnásobný DMRS symbol. Znázorněno na obr. 1.6.
- Typ 1 – Je zobrazen na prvním obrázku na obr. 1.7. Z něco vyplývá, že jsou definovány jednonásobné DMRS symboly. Tato konfigurace umožňuje pouze 2 možné kombinace umístění DMRS symbolů.
- Typ 2 – Zobrazuje dvojnásobné DMRS symboly. Má možnost 3 kombinací, kam umístit tyto sousedící symboly, což je více než u Typu 1. Podporuje tedy větší počet ortogonálních signálů, což je vhodné při použití multi-user MIMO. Konfigurace je znázorněna na druhém obrázku na obr. 1.7.



Obr. 1.5: Mapovací typ A [1, PDSCH DMRS]



Obr. 1.6: Mapovací typ B [1, PDSCH DMRS]



Obr. 1.7: Typ 1 a Typ 2 [1, PDSCH DMRS]

Phase Tracking Reference Signal (PTRS)

PTRS slouží k minimalizaci fázového šumu oscilátoru, který se zvyšuje spolu se zvyšující se pracovní frekvencí (což je důležité zejména pro oblast milimetrových vln, např. pro 5G NR ve využívaném pásmu 28 GHz). Jeho hlavní funkcí je sledování fáze lokálního oscilátoru na vysílači a přijímači. PTRS je přítomný v DL i v UL. PTRS má větší hustotu v časové doméně než ve frekvenční vzhledem k vlastnostem fázového šumu.

Sounding Reference Singal (SRS)

SRS je přenášen od UE k základnové stanici (Next generation NodeB (gNB)), které pomáhá získat Channel State Information (CSI) pro každého uživatele. CSI popisuje jak se signál šíří od uživatele k základnové stanici. Tzn. jak ubývá energie signálu s rostoucí vzdáleností a jakým způsobem jej ovlivňuje rozptyl. SRS tedy

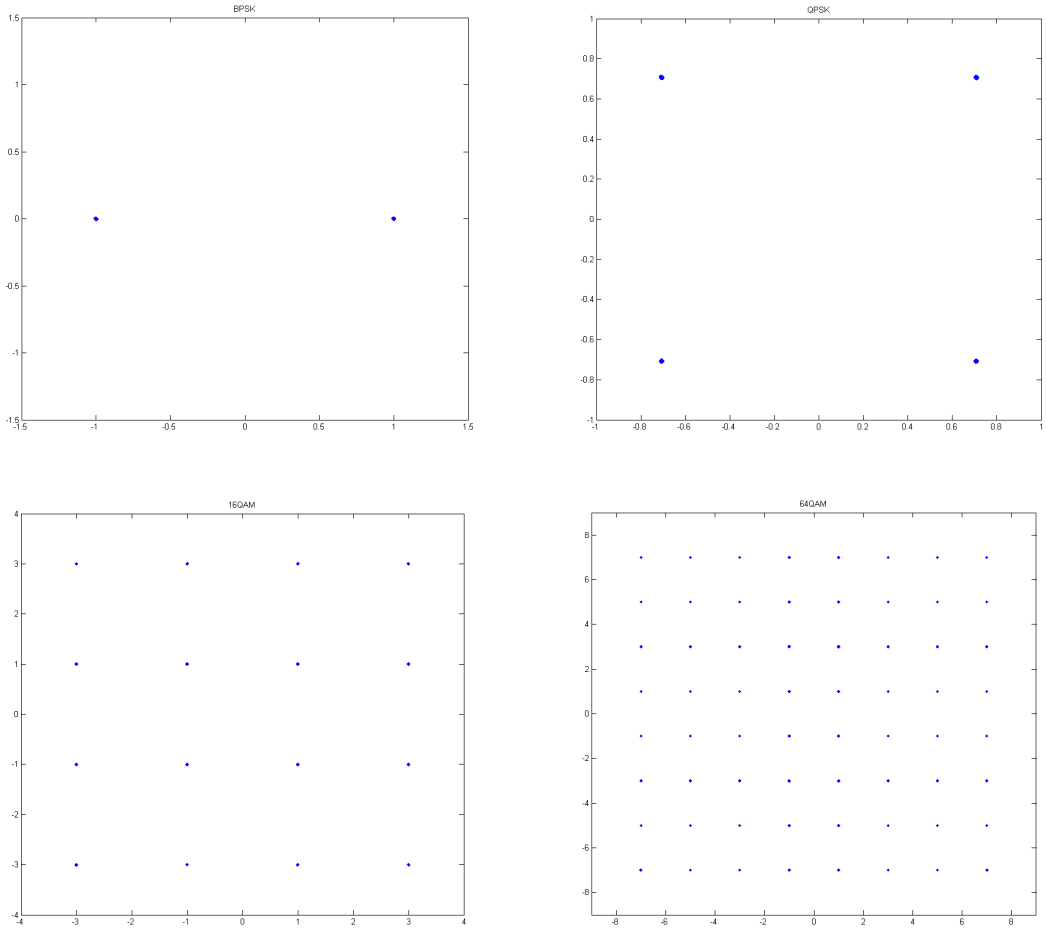
pomáhá v plánování zdrojů, adaptaci spojení, masivním MIMO a správě anténních svazků.

Channel State Information Reference Signal (CSIRS)

CSIRS je přenášen od základnové stanice k uživateli, který na základě tohoto signálu odhadne parametry rádiového kanálu a vypočítá CSI, které je poté v UL odesíláno základnové stanici.

1.1.7 Modulace

Podporované modulace jsou $\pi/2$ BPSK, BPSK, QPSK, 16QAM, 64QAM, 256QAM. Konstelační diagramy vybraných modulací lze vidět na obr. 1.8.



Obr. 1.8: Konstelační diagramy modulací (BPSK, QPSK, 16QAM, 64QAM)

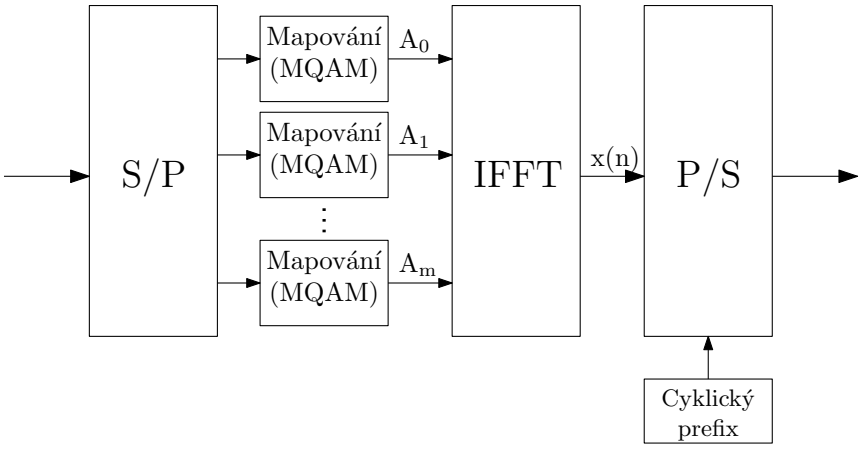
1.1.8 OFDM

Princip Orthogonal Frequency Division Multiplexing (OFDM) spočívá v rozdělení vstupního datového toku do paralelních větví, kde každá větev je mapována podle zvolené modulace a modulována na subnosný kmitočet. Výstupy jsou pak sečteny. Díky tomu, že je signál rozdělen na pomalejší paralelní větve a dojde k prodloužení doby trvání symbolu, je signál odolnější vůči vícecestnému šíření a úzkopásmovému rušení. Ortogonalita zajišťuje, že se subnosné navzájem neovlivňují. Pro OFDM modulátor pak využíváme Inverse Fast Fourier Transform (IFFT). Matematický popis OFDM modulátoru za pomoci IFFT je [5]

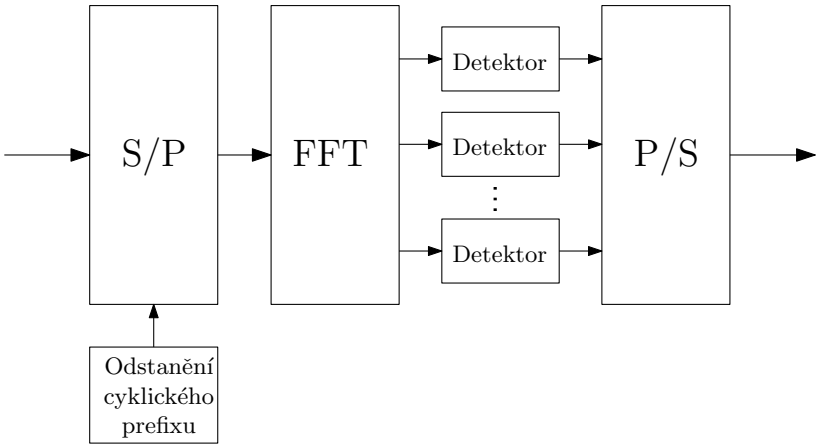
$$x(n) = \frac{1}{M} \sum_{m=0}^{M-1} A_m \cdot \mathrm{e}^{j\frac{2\pi mn}{M}} \tag{1.6}$$

kde m je pořadí subnosné, n je pořadí symbolu a M je velikost IFFT. Na obr. 1.9 a obr. 1.10 je vidět základní zapojení OFDM modulátoru a demodulátoru s použitím IFFT, resp. Fast Fourier Transform (FFT).

Datový tok přivádíme na vstup sériově-paralelního převodníku, který nám rozdělí vstupní bity na M paralelních větví (M subnosných). Ty dále vstupují do bloku IFFT, který převede signál z frekvenční do časové oblasti. Výstup IFFT pokračuje do paralelně-sériového převodníku, který vzorky převede na sériovou posloupnost a vloží mezi ně CP. CP zaručuje větší odolnost vůči vícecestnému šíření, které způsobuje mezisymbolové přeslechy (ISI). Takto získaná data jsou v komplexním tvaru. Pro vysílání je možné použít kvadraturní modulátor. OFDM demodulátor pracuje analogicky.



Obr. 1.9: OFDM modulátor



Obr. 1.10: OFDM demodulátor

1.1.9 Vyrovnávače kanálu

Vyrovnávače kanálu slouží ke kompenzaci jeho frekvenční charakteristiky, která z důvodu vícecestného šíření není konstantní pro všechny frekvence. Pak tedy dochází pro určité frekvence k zeslabení přenášeného signálu. Příklad přenosové charakteristiky kanálu a vyrovnávací funkce kanálu můžeme vidět na obr. 1.11. Příklad byl vygenerován impulzní charakteristikou `h=[1 zeros(1,6) 0.2]` pro dvě cesty šíření signálu s počtem subnosných 100. První cesta je přímá, druhá je zpožděná o 6 vzorků s výkonem sníženým na dvě desetiny původního výkonu.

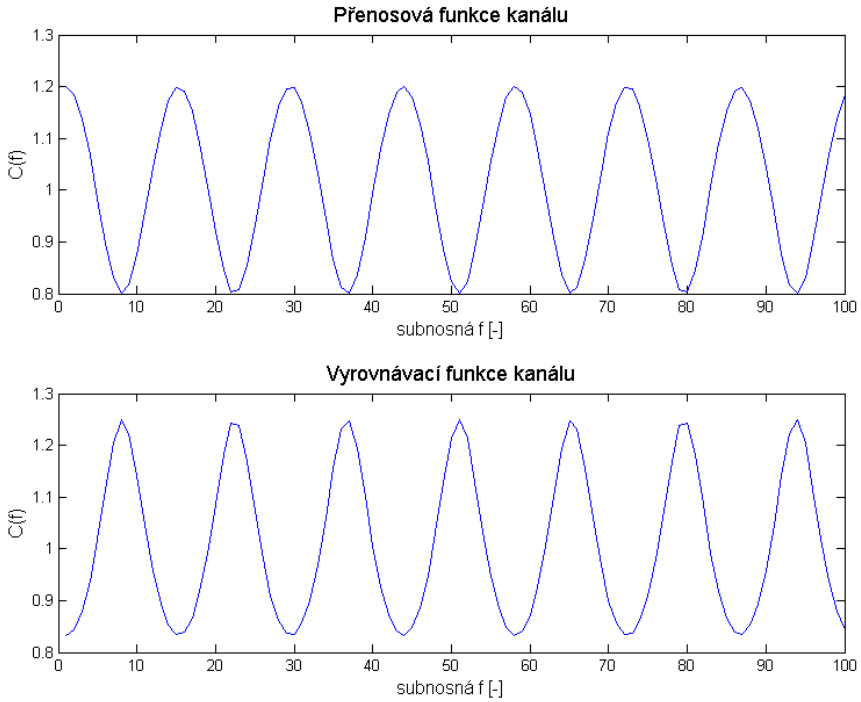
Vyrovnávače mohou být lineární nebo nelineární. Dále mohou být vyrovnávače s charakteristikou proměnnou v čase. Mezi nejjednodušší vyrovnávače patří tzv. Zero Forcing vyrovnávač, který byl použit při simulaci v programu Matlab.

Zero forcing vyrovnávač (který patří mezi lineární vyrovnávače) je v podstatě filtr, jehož přenosová charakteristika je inverzní k přenosové charakteristice kanálu. Přenosovou charakteristiku kanálu je možné určit z referenčních dat, která jsou známá jak ve vysílači tak v přijímači. Pro 5G můžeme využít výše popsaných DMRS. Předpokládejme dále, že vysílač vysílá DMRS se symboly v kmitočtové oblasti A_m, kde „m“ je index subnosné. Způsob odhadu přenosové charakteristiky kanálu pak lze zapsat

$$H_m = \frac{A'_m}{A_m}, \tag{1.7}$$

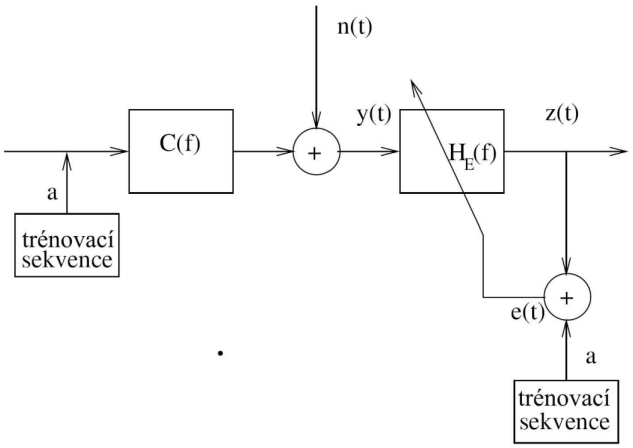
kde A'_m jsou přijaté DMRS symboly na nosných m = 0, 1, ... M-1. Symboly na výstupu vyrovnávače Y_m je pak možno zapsat dle vztahu

$$Y_m = X'_m \cdot \frac{1}{H_m} \tag{1.8}$$



Obr. 1.11: Přenosová a vyrovnávací funkce kanálu

kde X'_m jsou přijaté datové symboly na nosných m = 0, 1, ... M-1. Je třeba dodat, že pokud dojde ke změně přenosového kanálu, je třeba provést výpočet vyrovnávací funkce znova.



Obr. 1.12: MMSE vyrovnávač [12, Vyrovnavače kanálu]

Mezi lineární vyrovnávače patří také adaptivní Minimum Mean Square Error

(MMSE) vyrovnávač. Je založen na adaptivní minimalizaci střední kvadratické chyby mezi trénovacím a opraveným signálem. Koncepce je vidět na obr. 1.12. U lineárních vyrovnávačů nastává problém pokud se přenosová charakteristika kanálu blíží nule. V případě převrácené hodnoty dostáváme neúměrně vysoké hodnoty a zesilujeme šum. V takovém případě je vhodně použít nelineární vyrovnávače. Takovým zástupcem je Decision Feedback Equalizer (DFE) vyrovnávač.

1.2 ZYNQ Ultra Scale

Pro implementaci generátoru 5G signálů byl zvolen vývojový kit firmy Xilinx s označením Zynq UltraScale+ RFSoC ZCU111. Tento vývojový kit je přímo určen pro návrh a vývoj vysoce výkonných RF aplikací. Kit obsahuje vývojovou desku s osazeným čipem XCZU28DR-2FFVG1517E, propojovací kabely, filtry (2 x 2,5 GHz a 2 x 1,3 GHz LP, 2 x (3 – 4,3) GHz BP) a přídavnou desku, která obsahuje konektory umožňující propojení například do zpětné smyčky (loopback). Čip se skládá z Processing System (PS) a Programmable Logic (PL). PS zahrnuje 4 jádrový Arm Cortex-A53 a 2 jádrový real-time Arm Cortex-R5. PL obsahuje spoustu komunikačních rozhraní jako jsou například USB 3.0, SATA M.2, UART, SPI spolu s integrovanou DDR4 pamětí. Obsahuje také integrované ADC a DAC převodníky, které jsou schopny generovat a přijímat signály v jednom z přidělených frekvenčních pásem pro 5G a to 3,4 GHz – 3,8 GHz. Převodníky jsou součástí RFDC, který bude popsaný v kapitole 1.2.1. Vývojový kit jako takový pak obsahuje mimo jiné konektory pro výše uvedené komunikace a převodníky, patice pro DDR4 SODIMM paměti, integrované obvody TI LMK04208 a TI LMX2594RHAT pro generaci hodinového signálu a spoustu jiných.

Na obr. 1.13 jsou varianty čipů dostupných na Zynq UltaScale+ RFSoC a jejich parametry. Vyznačen je typ čipu, který je osazen na zmíněném kitu.

	Device Name	ZU21DR	ZU25DR	ZU27DR	ZU28DR	ZU29DR	ZU39DR	ZU43DR	ZU46DR	ZU47DR	ZU48DR	ZU49DR
				Gen 1			Gen 2	Gen 3				
PS		**Quad-core** Arm® Cortex™-A53 MPCore™ up to 1.3GHz, **Dual-core** Arm Cortex-R5 MPCore up to 533MHz										
RF Data Converter	12-bit RF-ADC # of ADCs	0	8	8	8	16	16	–	–	–	–	–
	12-bit RF-ADC w/DDC Max Rate (GSPS)	0	4.096	4.096	4.096	2.058	2.220	–	–	–	–	–
	14-bit RF-ADC # of ADCs	–	–	–	–	–	–	4	8 4	8	8	16
	14-bit RF-ADC w/DDC Max Rate (GSPS)	–	–	–	–	–	–	5.0	2.5 5.0	5.0	5.0	2.5
	14-bit RF-DAC # of DACs	0	8	8	8	16	16	4	12	8	8	16
	14-bit RF-DAC w/DUC Max Rate (GSPS)	0	6.554	6.554	6.554	6.554	6.554	10.0	10.0	10.0	10.0	10.0
	SD-FEC	8	0	0	8	0	0	0	8	0	8	0
	Number of DDCs per RF-ADC(1)	0	1	1	1	1	1	2	1	1	1	1
	RF input Freq max. GHz			4			5			6		
	Decimation / Interpolation			1x, 2x, 4x, 8x			1x, 2x, 4x, 8x	1x, 2x, 3x, 4x, 5x, 6x, 8x, 10x, 12x, 16x, 20x, 24x, 40x				
Programmable Logic (PL)	System Logic Cells (K)	930	678	930	930	930	930	930	930	930	930	930
	CLB LUTs (K)	425	310	425	425	425	425	425	425	425	425	425
	Max. Dist. RAM (Mb)	13.0	9.6	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0
	Total Block RAM (Mb)	38.0	27.8	38.0	38.0	38.0	38.0	38.0	38.0	38.0	38.0	38.0
	UltraRAM (Mb)	22.5	13.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5	22.5
	DSP Slices	4,272	3,145	4,272	4,272	4,272	4,272	4,272	4,272	4,272	4,272	4,272
	GTY Transceivers	16	8	16	16	16	16	16	16	16	16	16
	PCIe® Gen3 x16	2	1	2	2	2	2	–	–	–	–	–
	PCIeGen3 x16/Gen4 x8 / CCIX(2)	–	–	–	–	–	–	2	2	2	2	2
	150G Interlaken	1	1	1	1	1	1	1	1	1	1	1
	100G Ethernet MAC/PCS w/RS-FEC	2	1	2	2	2	2	2	2	2	2	2
	System Monitor	1	1	1	1	1	1	1	1	1	1	1
	Speed Grades	-1E, -1I, -1LI, -2E, -2LE, -2I, -2LI	-1E, -1I, -1LI, -2E, -2LE, -2I, -2LI	-1E, -1I, -1LI, -2E, -2LE, -2I, -2LI	-1E, -1I, -1LI, -2E, -2LE, -2I, -2LI	-1E, -1I, -1LI, -2E, -2LE, -2I, -2LI	-2I, -2LI	-1E, -1I, -1LI, -2E, -2I, -2LI	-1E, -1I, -1LI, -2E, -2I, -2LI	-1E, -1I, -1LI, -2E, -2I, -2LI	-1E, -1I, -1LI, -2E, -2I, -2LI	-1E, -1I, -1LI, -2E, -2I, -2LI

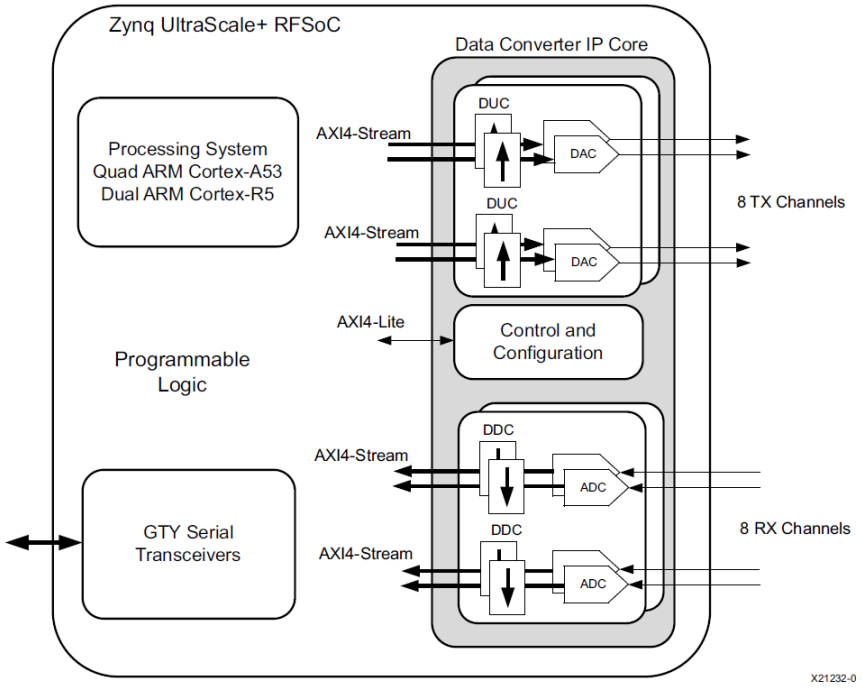
Obr. 1.13: Přehled parametrů čipů dostupných na Zynq UltaScale+ RFSoC [8]

Vývojový kit může pracovat ve 3 módech:

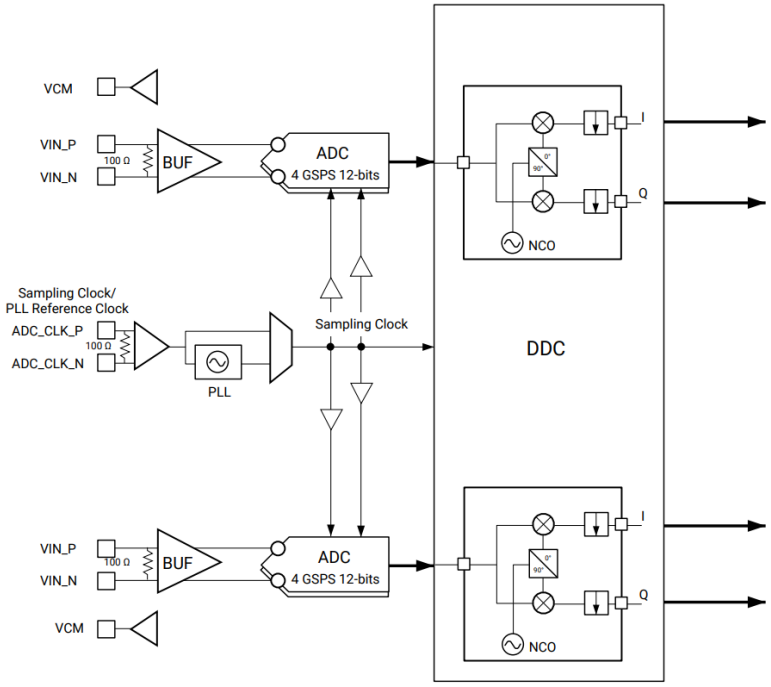
- Samostatné DAC – Pomocí GUI jsou na PC generována data, která jsou posílána na konkrétní DAC. Výstup DAC pak může být monitorován externím vybavením (osciloskop, spektrální analyzátor).
- Samostatné ADC – Externě generovaný signál je přiveden na vstup konkrétního ADC. Přijatá digitální data mohou být dále zpracována např. na PC.
- Loopback mezi DAC a ADC – Výstup DAC je smyčkou přiveden zpět do kitu na ADC.

1.2.1 RF Data Converter

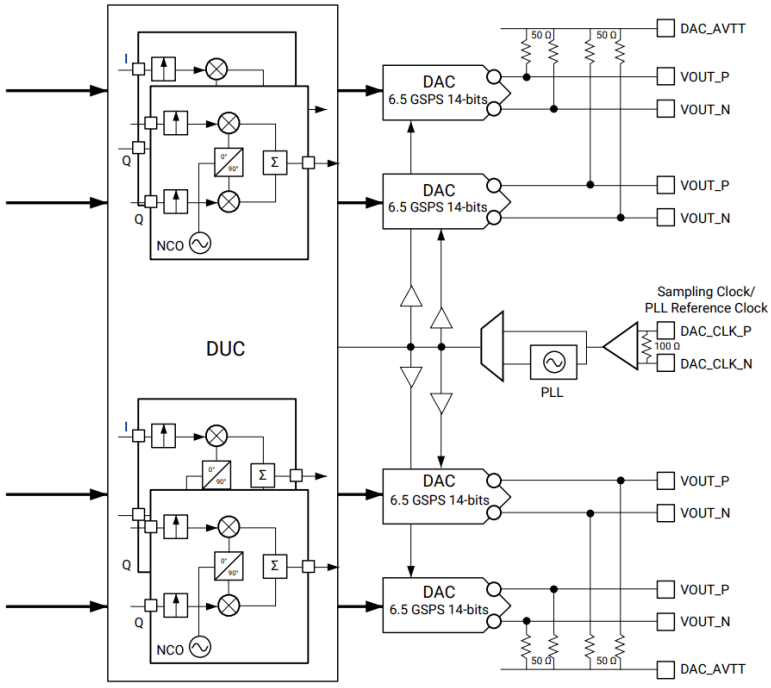
Jednou z klíčových částí čipu je RFDC (obr. 1.14). RFDC obsahuje osm DAC a osm ADC kanálů. DAC má rozlišení 14 bitů a maximální vzorkovací frekvenci 6,554 GS/s. ADC má rozlišení 12 bitů a maximální vzorkovací frekvenci 4,096 GS/s. DAC a ADC jsou organizovány do dlaždic (tiles). Každá dlaždice obsahuje čtyři DAC a dva nebo čtyři ADC, kde každá dlaždice má samostatný blok s PLL. Dále RFDC obsahuje Digital Up Converter (DUC) (obr. 1.16) a Digital Down Converter (DDC) (obr. 1.15) pro každý převodník.



Obr. 1.14: RF Data Converter [10, RFDC LogiCORE IP]



Obr. 1.15: Struktura ADC+DDC [10, RFDC LogiCORE IP]



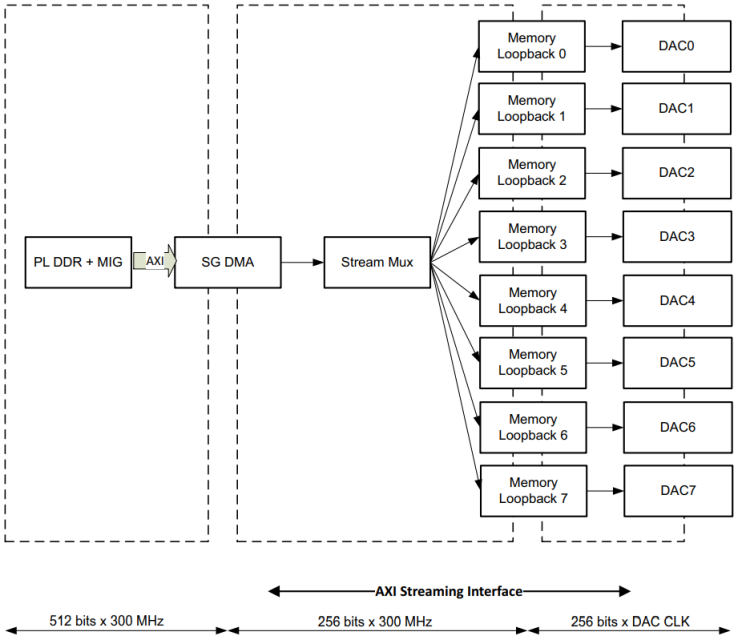
Obr. 1.16: Struktura DAC+DUC [10, RFDC LogiCORE IP]

DUC a DDC obsahují bloky pro programovatelnou interpolaci/decimaci, jemný směšovač (Fine Mixer) s numericky řízeným oscilátorem (Numerical Controlled Oscillator (NCO)) a hrubý směšovač (Coarse Mixer). Jemný směšovač používá NCO s registrem o délce 48 bitů pro nastavení směšovací frekvence v platném rozsahu frekvencí od $-\frac{F_s}{2}$ do $\frac{F_s}{2}$ pro maximální hodnotu frekvence ±10 GHz. Hrubý směšovač je schopen směšovat s frekvencemi rovnými $\frac{F_s}{2}$, $\frac{F_s}{4}$ a $-\frac{F_s}{2}$. Převodníky jsou umístěny v RFDC IP jádru, který je propojen s ostatními částmi FPGA pomocí AXI4 sběrnice.

1.2.2 RFDC tok dat

Tok dat u DAC Na obr. 1.17 je zobrazeno 8 kanálů RF-DAC. Paměťové smyčky (Memory Loopback) jsou použity pro opakování signálu, protože rychlost RF-DAC je velmi vysoká a není tak možné přenášet data v reálném čase. Výstup DAC má tak na výstupu stále nějaký signál. Možnosti pro uložení a přehrání vstupních dat jsou dvě:

- Vzorky jsou uloženy v PL DDR, která má velkou kapacitu (4 GB), ale malou rychlost (2666 MT/s).
- Vzorky jsou uloženy v blokové paměti RAM (uvnitř FPGA), která má malou kapacitu (cca 38 Mb), ale velkou rychlost (v závislosti na nastaveném hodinovém signálu).

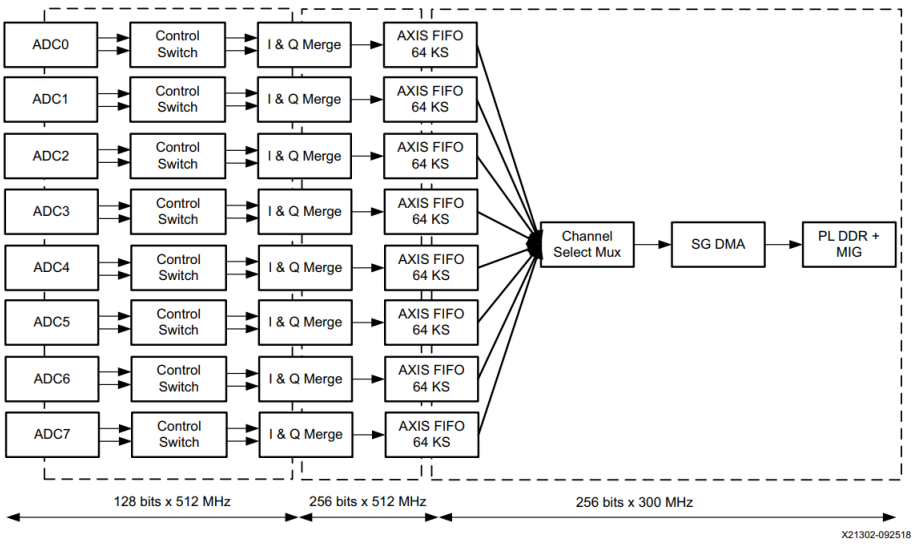


Obr. 1.17: Tok dat u DAC

Subsystém DAC dále obsahuje Scatter Gather DMA (SG DMA), který slouží pro lokalizaci dat v paměti, které budou vyslány do DAC. DMA pak tyto data posílá do MUX bloku. který na základě vybraného kanálu odesílá data do odpovídajícího DAC.

Tok dat u ADC Ovládací spínač (Control Switch) z obr. 1.18 určuje, zda je vstupní tok ADC poslán dále. Pokud ano, data vstupují do I&Q Merge logiky, odkud podle uživatelského nastavení pokračují do AXIS FIFO buď reálná nebo komplexní data. Multiplexer kanálů (Channel Select MUX) vždy propojí jeden z kanálů ADC (podle volby uživatele) k Direct Memory Access (DMA). Tyto data pokračují do SG DMA, aby mohla být uložena do paměti. Synchronizace všech kanálů je zajištěna logikou ovládacích přepínačů (Control Switch).

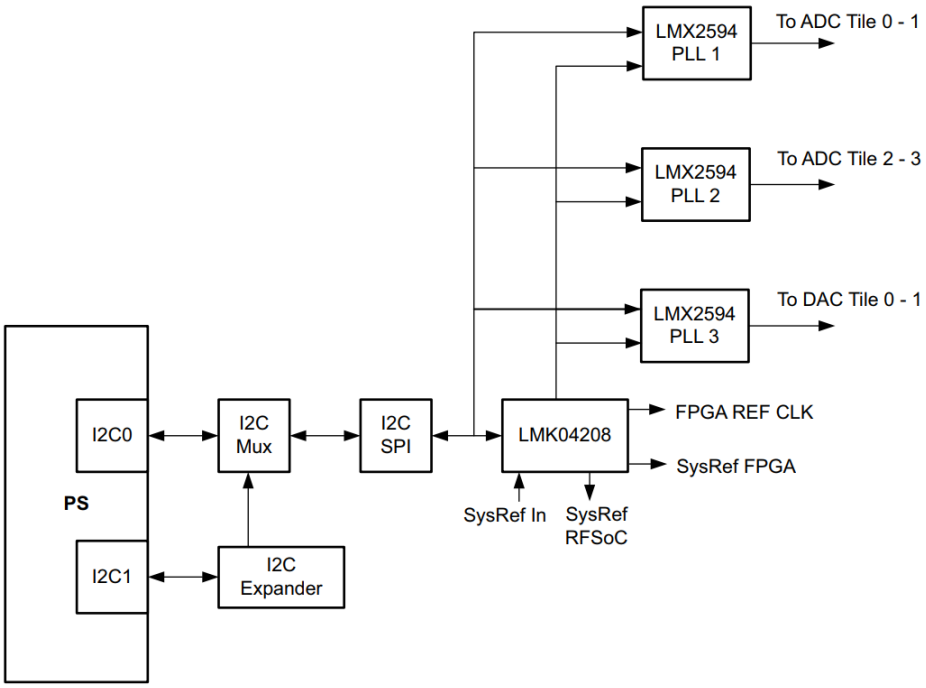
Pro komunikaci mezi paměťmi (PL-DDR) a RFDC v rámci programovatelné logiky (Programmable Logic) se používá AXI sběrnice. Jako komunikace systému zpracování (Processing System) se používá Gigabitový Ethernet, I2C a rozhraní pro SD kartu.



Obr. 1.18: Tok dat u ADC

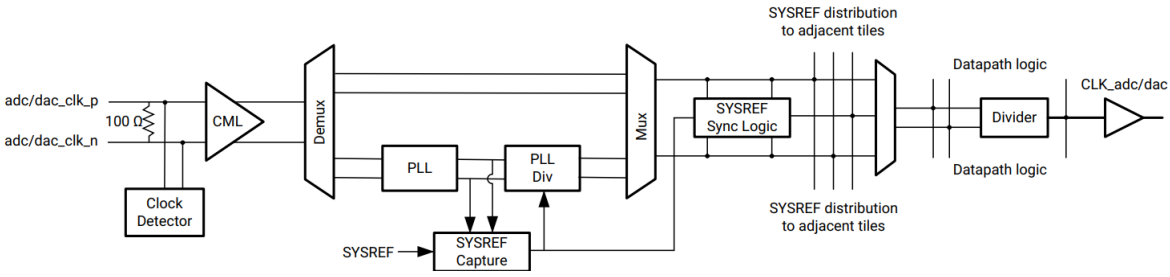
1.2.3 Hodinové signály a časování

RFDC je řízen hodinovým signálem, který je generován za pomoci primárních referenčních PLL (LMK04208) a RF PLL (LMX2594), které jsou umístěny na DPS ZCU111. RF PLL slouží ke generování vzorkovací frekvence pro DAC a ADC. Primární rozhraní pro konfiguraci těchto PLL je Serial Peripheral Interface (SPI), které je řízeno rozhraním Inter-Integrated Circuit (I2C) pomocí převodníku I2C na SPI. Popsaná struktura je vidět na obr. 1.19. Každá DAC nebo ADC dlaždice má svůj vlastní vstup pro hodinový signál. Navíc je tu dedikovaný vstup PL SYSREF, který slouží pro synchronizaci mezi dlaždicemi (multi-tile synchronization) nebo mezi čipy (multi-chip synchronization).



Obr. 1.19: Zdroje hodinového signálu ZCU111 [10, RFDC LogiCORE IP]

DAC i ADC dlaždice obsahují hodinovou strukturu se vstupním děličem, PLL a výstupním děličem. Hodinový signál vstupující do dlaždice může být použit jako zdroj hodinového signálu pro převodníky, nebo může sloužit jako reference pro PLL v dlaždici. Výstup PLL může pak sloužit jako vzorkovací frekvence a může být použit i jako vstup pro SYSREF. DAC i ADC má potom lokální děličku, kterou může dělit hodinový signál z PLL. Popsaná struktura je vidět na obr. 1.20

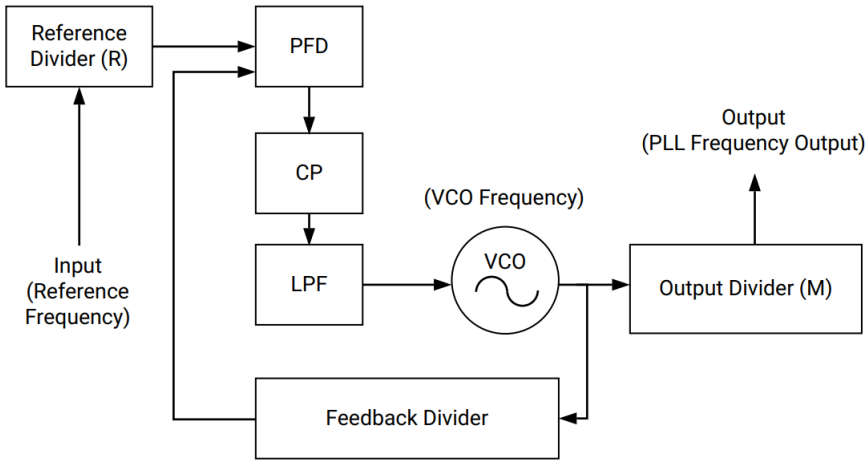


Obr. 1.20: Struktura hodinových domén v dlaždici [10, RFDC LogiCORE IP]

Parametry PLL se konfigurují v IP jádru RFDC. PLL musí být povoleno, pokud chceme za běhu aplikace měnit paramtry PLL pomocí API. Na následujícím obrázku (obr. 1.21) je vidět blokový diagram struktury PLL v dlaždici. Děličkami - Reference Divider (R), Feedback Divider (FBDiv) i Output Divider (M) se nastavuje požadovaná frekvence, přičemž nejnižšího fázového šumu je dosaženo, pokud $R = 1$. Výstupní frekvence PLL je dána vztahem [10]

$$f_{vz} = \frac{f_{in}}{R} \cdot \frac{FBDiv}{M} \tag{1.9}$$

kde f_{in} je referenční frekvence (Reference Frequency).

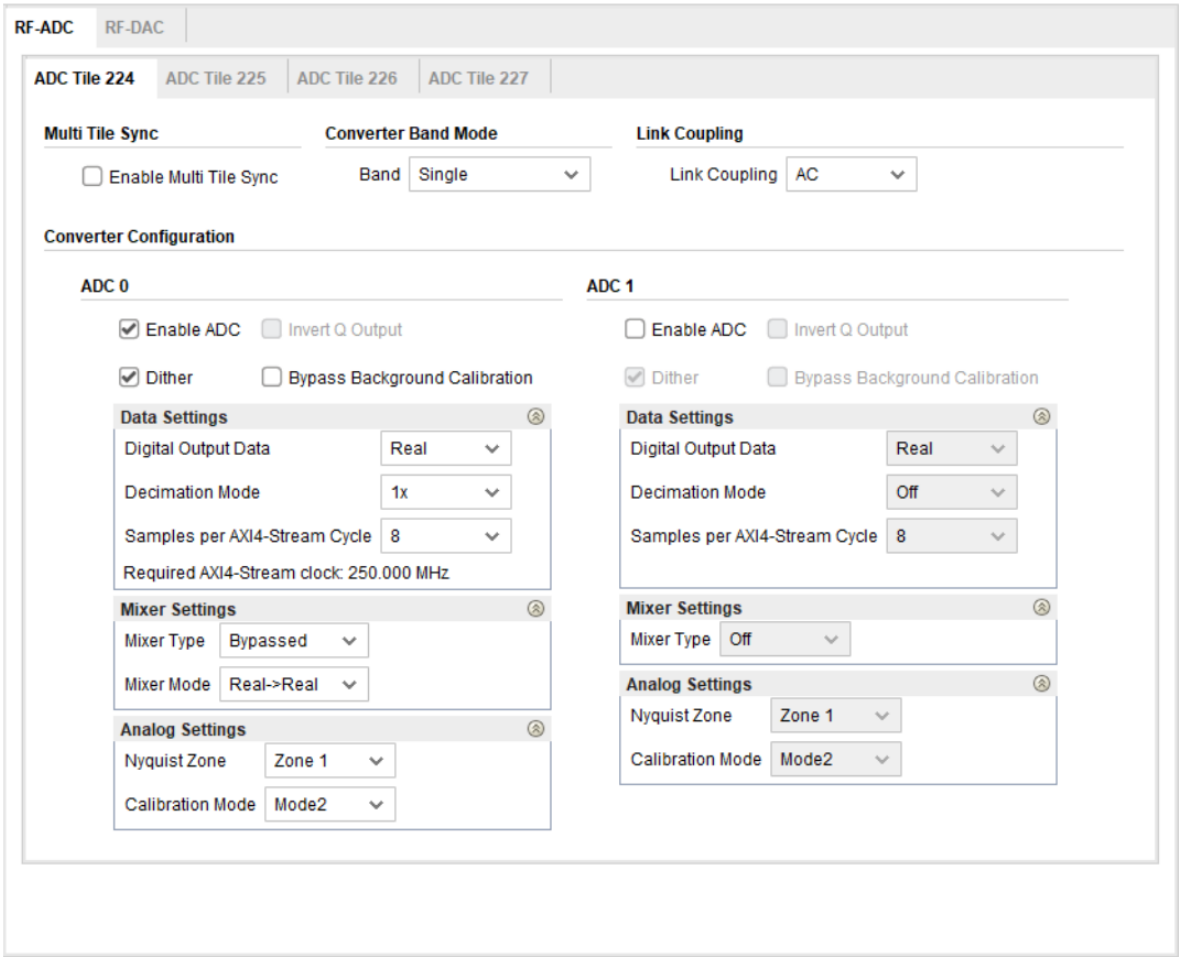


Obr. 1.21: Struktura PLL [10, RFDC LogiCORE IP]

1.2.4 Možné způsoby konfigurace RFDC

Vývojové prostředí Vivado firmy Xilinx obsahuje IP integrátor, který umožňuje vytvořit návrh hardwaru, který je rozdělen mezi PS, RFDC a PL. PS je konfigurován pomocí Ethernetového a I2C kontroléru. Pomocí Ethernetu můžeme komunikovat mezi počítačem a kitem, I2C zajišťuje komunikaci mezi PS a RF PLL. SD karta obsahuje konfiguraci, která zajistí správné načtení FPGA po připojení napájení. Data odesílaná přes Ethernet jsou ukládána do PL DDR pomocí DDR kontroléru. Aplikace běžící na procesoru pak přenese tyto data do programovatelné logiky přes AXI sběrnici.
Spojení a komunikace s vývojovým kitem může být uskutečněna dvěma způsoby:

- S použitím GUI (RFDC Evaluation Tool a RF Analyzer) – Obě uživatelská rozhraní se používají pro řízení a analýzu vývojového kitu. RFDC Evaluation Tool slouží jako nástroj pro konfiguraci a testování RFDC. Umožňuje konfiguraci hodinového signálu, jednotlivých ADC a DAC, odeslání a příjem vybraných dat. Slouží také jako debuger pro RF Analyzer. RF Analyzer je komplexnější. Pro řízení používá RF Analyzer mikroprocesor (MicroBlaze), který je implementován v PL. MicroBlaze přijímá příkazy od RF Analyzeru přes JTAG rozhraní a komunikuje pomocí AXI sběrnice se zbytkem PL, která odpovídá našemu návrhu z IP konfigurace. Rozhraní pracují s formáty souborů .lvm a .tdms. Data, která chceme vysílat tedy musíme nejdříve vygenerovat např. v programu Matlab, poté překonvertovat na jeden z těchto formátů a pak umístit do adresáře s uživatelským rozhraním, odkud jej zvolíme z GUI.
- Bez použití GUI – Ve své podstatě je tento způsob podobný funkci RF Analyzátoru. Za pomoci IP generátoru ve Vivadu by mělo být možné vygenerovat příslušné bloky a propojení. Procesor uvnitř PL by s řídícím počítačem komunikoval pomocí Ethernetu, UARTu nebo jiného rozhraní, zatímco pro komunikaci se zbytkem PL by sloužila sběrnice AXI. Počítač tedy data odešle do kitu a integrovaný procesor je pomocí AXI sběrnice uloží do zvolené paměti (BRAM/DDR). RFDC si poté data načte, zpracuje a vyšle. Příjem probíhá analogicky.



Obr. 1.22: Okno RFDC IP jádra pro nastavení ADC [10, RFDC LogiCORE IP]

Konfigurace IP jádra

Pro správnou funkci je třeba RFDC správně nakonfigurovat. Konfigurační okno IP jádra pro ADC je na obr. 1.22. Soupis nastavení je pak v tab. A.1.
Možnosti nastavení ADC dlaždice (ADC tile):

- Link Coupling - Určuje, zda vstupní signál do RF-ADC je se střídavou nebo stejnosměrnou vazbou (AC/DC-coupled). V našem případě použijeme střídavou vazbu (AC-coupling).
- Converter Band Mode - Každý převodník může pracovat buď sám (single band mode) nebo s dalšími převodníky v dlaždici (multi-band mode). Touto možností vybíráme, který mód operace se bude používat. V našem případě si vystačíme s jedním převodníkem, tzn. single band mode.

Možnosti nastavení ADC (ADC n):

- Enable ADC - Určuje, zda vybraný převodník v odpovídající dlaždici je povolen nebo zakázán. V našem případě můžeme zvolit jakýkoliv, nemělo by záležet na tom který.
- Invert Q Output - Tento parametr je možné konfigurovat pouze, pokud jsou výstupní data RFDC I/Q a je povolen jemný směšovač. Pokud je to splněno, je možné pak generovat -Q data na výstupu. Tuto možnost nebudeme používat.
- Dither - Zvolí, zda je ve vybrané dlaždici povolen nebo ne. Dither by měl být povolen pokud vzorkovací frekvence je pod 3 GHz pro 4 GS/s RF-ADC (pod 1.5 GHz pro 2 GS/s variantu).
- Bypass Background Calibration - Vybráním této možnosti říkáme, zda má být do IP jádra implementována logika pro kalibrace na pozadí. Tato možnost je povolena pouze při reálném vstupu a reálném výstupu. V našem případě neaktivujeme tuto možnost, protože bychom měli mít reálný vstup a I/Q výstup.

Možnosti nastavení dat ADC (Data Settings):

- Digital Output Data - Zvolí typ dat vybraného převodníku v dlaždici. Validní data jsou reálná nebo I/Q. Když je převodník 0 nebo 2 nastaven na I/Q, musí být povolen i převodník 1, respektive 3. V opačném případě je konfigurace chybná. My chceme na výstupu převodníku I/Q data.
- Decimation Mode - Umožňuje nastavení decimace v hodnotách x1, x2, x4, x8.
- Samples per AXI4-Stream word - nastaví počet slov na cyklus. Platné hodnoty jsou mezi 1 a 8. V závislosti na nastavení se zobrazí hodinový signál AXI4-Stream sběrnice, který nesmí překročit 500 MHz. Tento hodinový signál vstupuje do IP jádra.

Možnosti nastavení směšovače ADC (Mixer Settings):

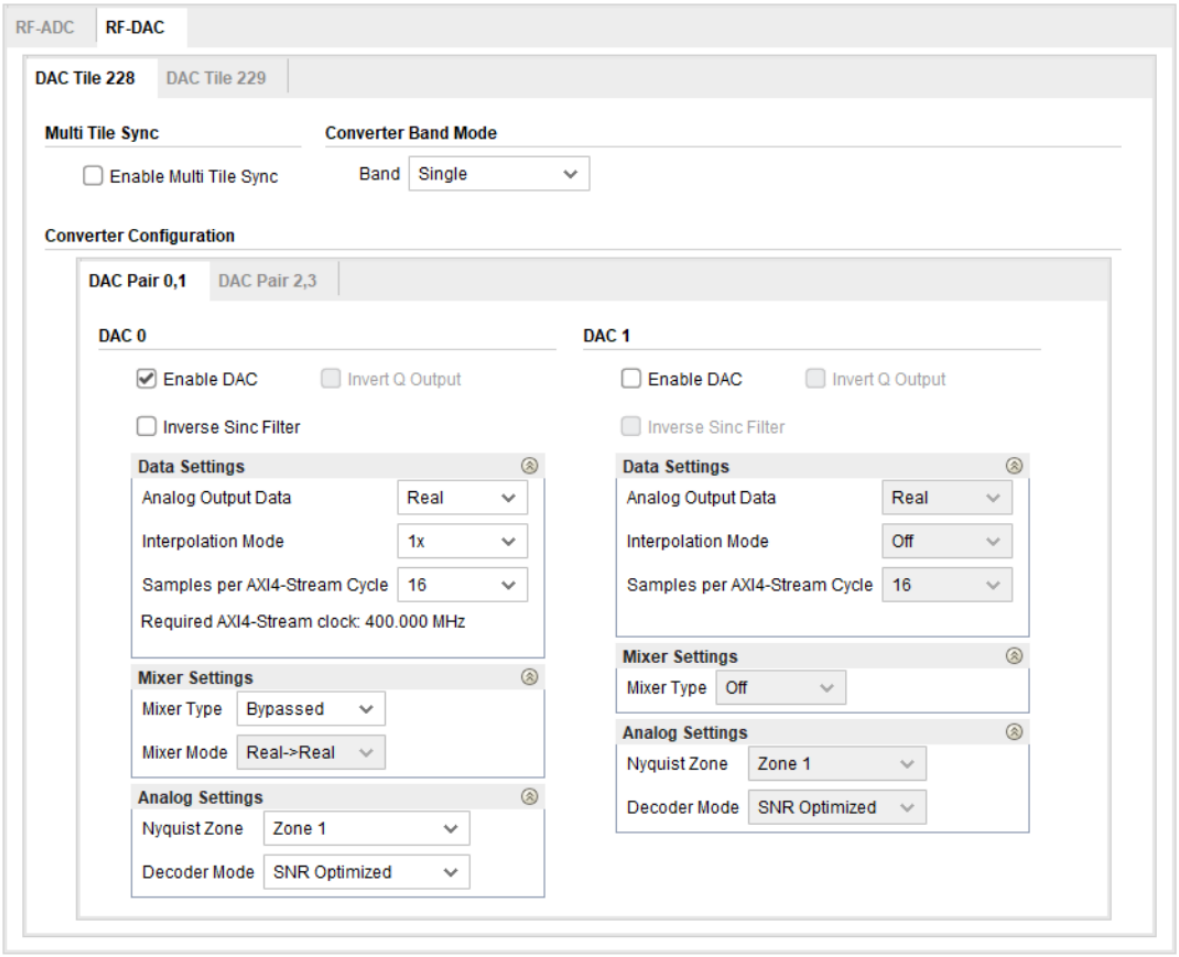
- Mixer Type - Umožňuje zvolit druh směšovače, který se má použít. Možnosti jsou žádný (bypassed), jemný (fine) a hrubý (coarse). My budeme používat jemný směšovač s NCO.
- Mixer Mode - Umožňuje zvolit mód směšovače pro vybraný převodník v dané dlaždici. Možnosti závisí na typu mixeru a vybraném formátu výstupních dat. Pokud jsou výstupem reálná data, mixer se nepoužívá. Pokud jsou výstupem I/Q data, mixer může být nastaven na Real to I/Q nebo I/Q to I/Q. V našem případě použijeme reálná data na I/Q data.
- Coarse Mixer Frequency - Nastaví frekvenci hrubého směšovače, pokud je povolen. Platné nastavení je $\frac{F_s}{2}$, $\frac{F_s}{4}$ a $-\frac{F_s}{4}$.
- Fine Mixer Frequency - Nastaví frekvenci jemného směšovače, pokud je povolen. Validní rozsah frekvencí je od $-\frac{F_s}{2}$ do $\frac{F_s}{2}$ pro maximální hodnotu frekvence

±10 GHz.

- Fine Mixer Phase - Nastaví fázi jemného směšovače, pokud je povolen. Validní rozsah je od $-180°$ do $180°$.

Možnosti nastavení ADC v analogové doméně (Analog Settings):

- Nyquist Zone - Volba mezi sudou a lichou Nyquistovou zónou.
- Calibration mode - Volba mezi rozdílnými druhy kalibračních optimalizačních schémat v závislosti na vlastnostech vstupních signálů. Mode 1 je optimální pro vstupní frekvence $\frac{F_s}{2}$ (Nyquist) ±10%. V jiných případech je vhodné použít Mode 2.



Obr. 1.23: Okno RFDC IP jádra pro nastavení DAC [10, RFDC LogiCORE IP]

Konfigurační okno IP jádra pro DAC je na obr. 1.23. Soupis nastavení je pak v tab. A.2.

Možnosti nastavení DAC dlaždice (DAC tile) (pouze parametry, které nejsou analogické s ADC):

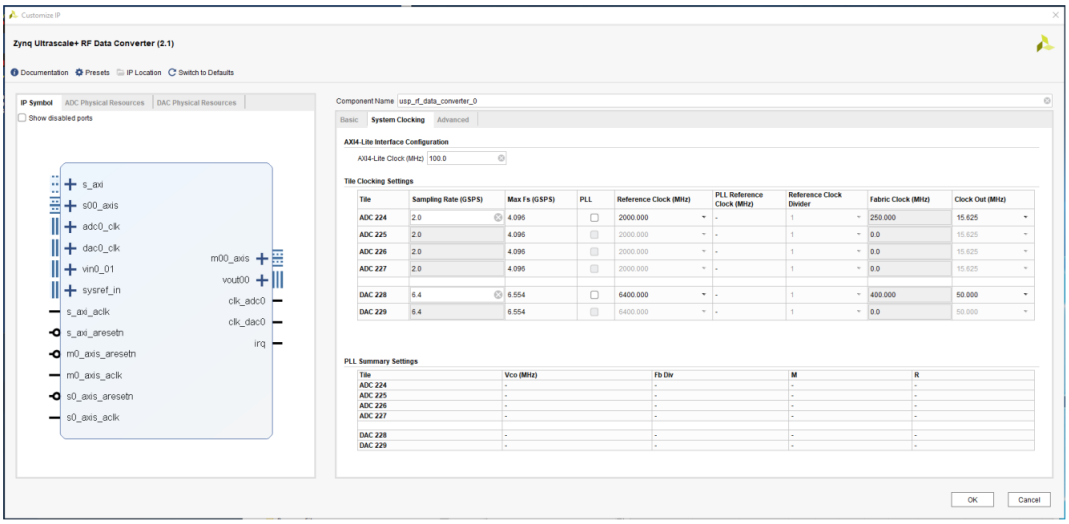
- Inverse Sinc Filter - Nastavení, zda má být sinc filtr zapnutý nebo ne. V našem případě ho povolíme.
- Analog Output Data - Zvolíme Real.

Možnosti nastavení dat DAC (Data Settings):

- Interpolation Mode - Umožňuje nastavení interpolace v hodnotách x1, x2, x4, x8.

Možnosti analogového nastavení DAC (Analog settings):

- Decoder mode - DAC může být optimalizován na nízké Signal-to-Noise Ratio (SNR) nebo na vysokou linearitu.



Obr. 1.24: Okno RFDC IP Coru pro nastavení hodin [10, RFDC LogiCORE IP]

Možnosti nastavení hodinového signálu dlaždice (Tile Clock Settings), na obr. 1.24:

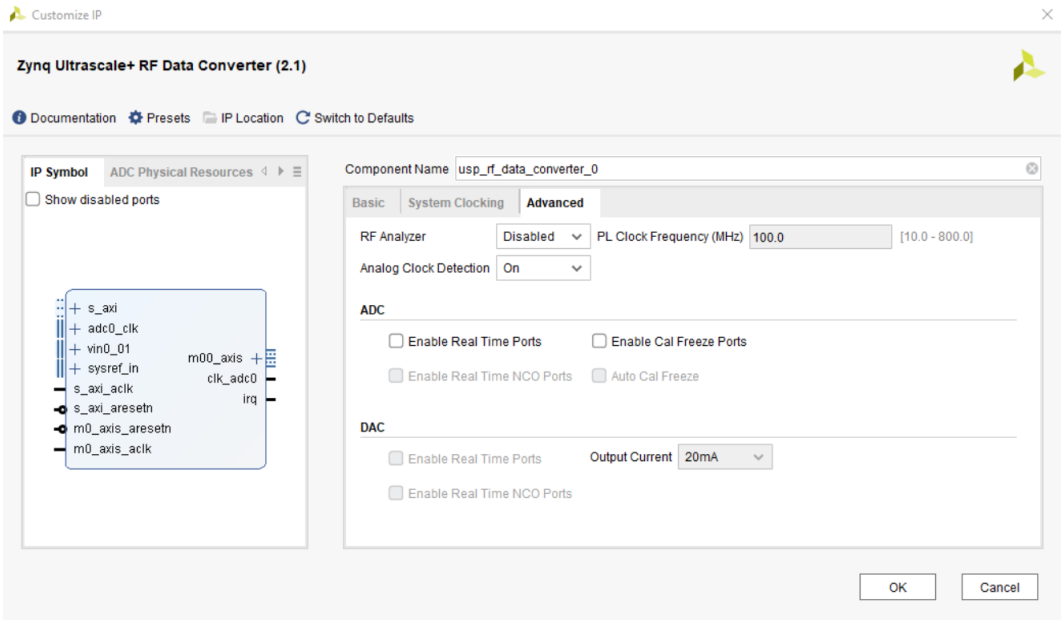
- Sampling Rate (GSPS) - Nastaví vzorkovací rychlost pro každou dlaždici.
- Max F_s (GSPS) - Maximální vzorkovací frekvence pro každou dlaždici (pouze pro informaci).
- PLL - Nastavuje, zda se PLL v rámci dlaždice používá nebo ne.
- Reference Clock (MHz) - Nastavuje frekvenci hodinového vstupu pro dlaždici. Hodnota závisí na vzorkovací frekvenci dlaždice.
- PLL Reference Clock - Nastavuje frekvenci hodinového signálu vstupujícího do PLL.
- Reference Clock Divider - Není určeno pro použití uživatelem. Vždy nastaveno na 1.
- Fabric Clock (MHz) - Hodinový signál učený pro AXI4-Steam pro vybranou dlaždici. Požadovaná frekvence je určena vzorkovací frekvencí a nastavením v

konfiguraci převodníku. Všechny porty AXI4- Stream sdílejí na dlaždici společný hodinový signál.

- Clock Out (MHz) - Nastavuje frekvenci výstupního hodinového signálu dlaždice. Lze ho například použít pro AXI4-Stream sběrnici. Hodnoty závisí na vzorkovací frekvenci dlaždice.

Souhrn nastavení PLL (PLL Summary Settings):

- VCO (MHz) - Frekvence napětím řízeného oscilátoru (Voltage Control Oscillator (VCO)).
- Fb Div - Nastavení zpětnovazebního děliče.
- M - Nastavení výstupního děliče.
- R - Nastavení děliče referenčního hodinového signálu.



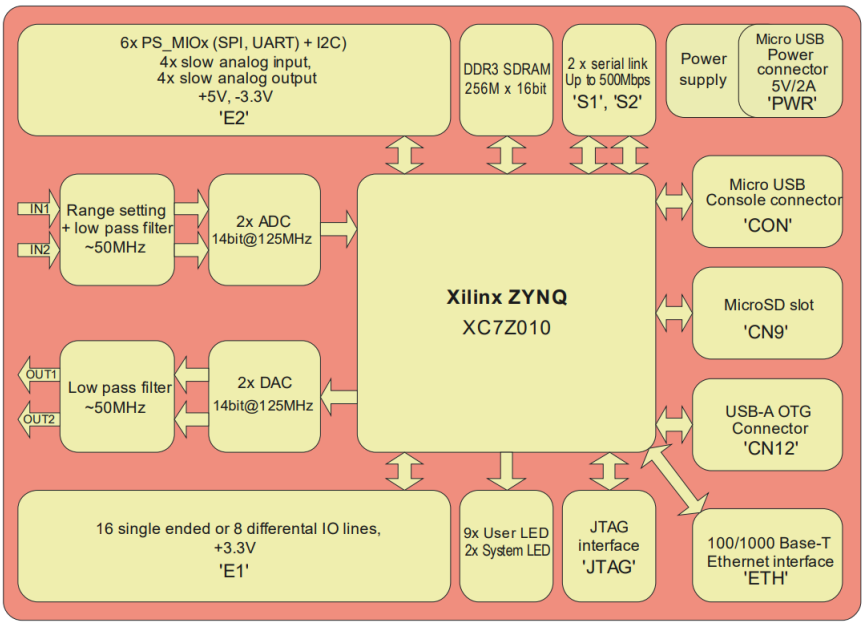
Obr. 1.25: Okno RFDC IP Coru pro pokročilé nastavení [10, RFDC LogiCORE IP]

Možnosti pokročilého nastavení (Advanced Settings), na obr. 1.25:

- RF Analyzer - Poskytuje hardwarový testovací systém. Systém obsahuje datové stimuly a zachycovací bloky nakonfigurované pro tuto instanci IP jádra.
- PL Clock Frequency (MHz) - Pro případ, kdy je vyžadována synchronizace při použití více dlaždic současně.
- Analog Clock Detection - Není dostupné pro uživatelskou konfiguraci.

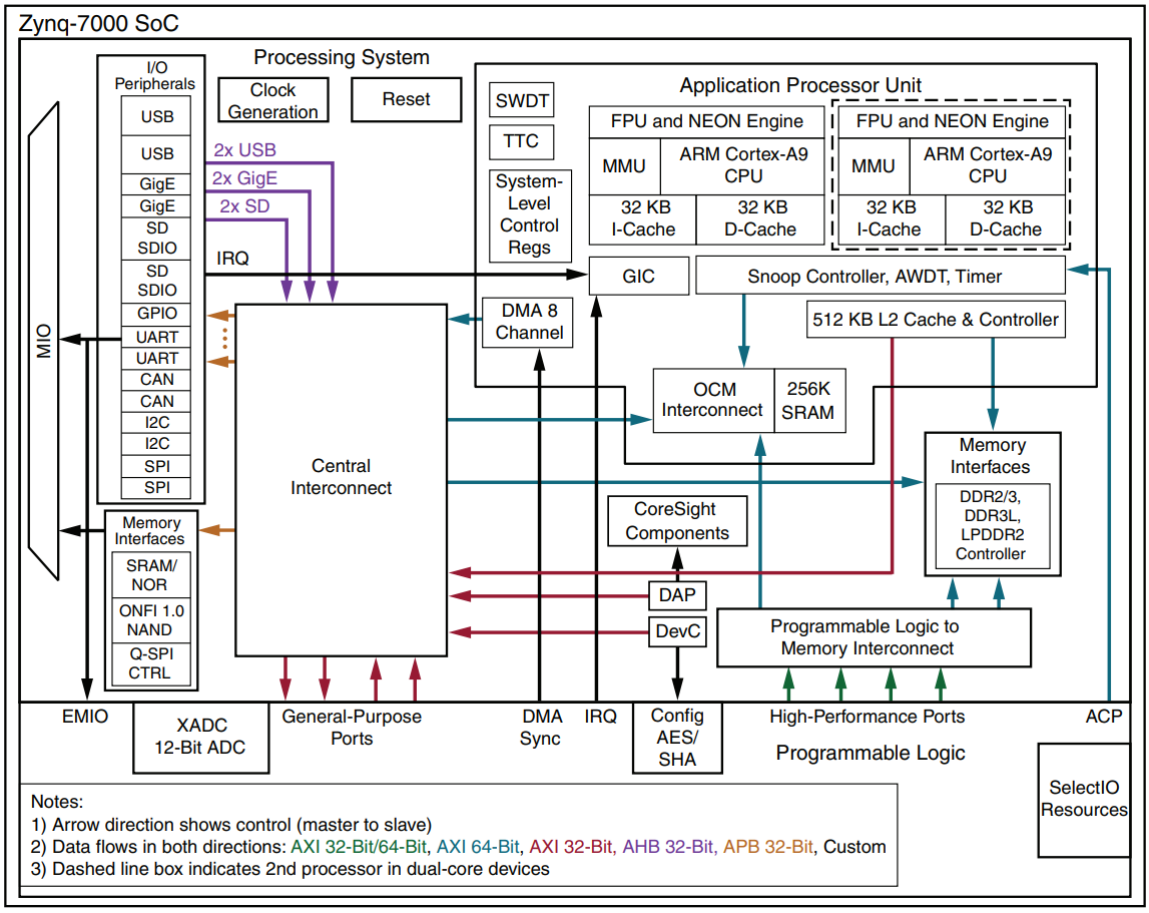
1.3 STEMLab Red Pitaya a Zynq XC7Z010

Red Pitaya (RP) je open source platforma, která byla vyvinuta, aby sloužila jako levná alternativa k drahým laboratorním měřicím přístrojům. Lze ji využívat například jako osciloskop, signální generátor nebo spektrální analyzátor. Kromě toho ji lze využít jako platformu pro jakékoliv uživatelské zadání, například v oblasti sdělovací techniky. Na obr. 1.26 je vidět bloková struktura RP. Vývojová deska obsahuje FPGA čip od firmy Xilinx s označením Zynq XC7Z010. Dále RP obsahuje USB a Etherenet konektor, SMA konektory pro RF vstupy a výstupy a slot pro micro SD kartu. Na SD kartě je umístěn OS Linux, který bootuje procesorem ARM Cortex-A9 vždy při připojení napájení. Dvoujádrový ARM Cortex-A9 je součástí FPGA čipu. Díky tomu je RP možno konfigurovat a programovat různými programovacími jazyky [17]. RP lze také snadno konfigurovat pomocí webového rozhraní [18].



Obr. 1.26: Blokové schéma vývojového kitu RedPitaya [16]

Vstupy a výstupy pro RF jsou připojeny přes dolní propust s mezním kmitočtem 50 MHz na DAC s označením DAC1401D125HL a ADC s označením LTC2145CUP-14. Oba převodníky mají maximální vzorkovací kmitočet 125 MSa/s s rozlišením 14 bitů [15]. Převodníky jsou připojeny k výše zmíněnému FPGA čipu. Na obr. 1.27 lze vidět funkční bloky čipů architektury Zynq-7000.

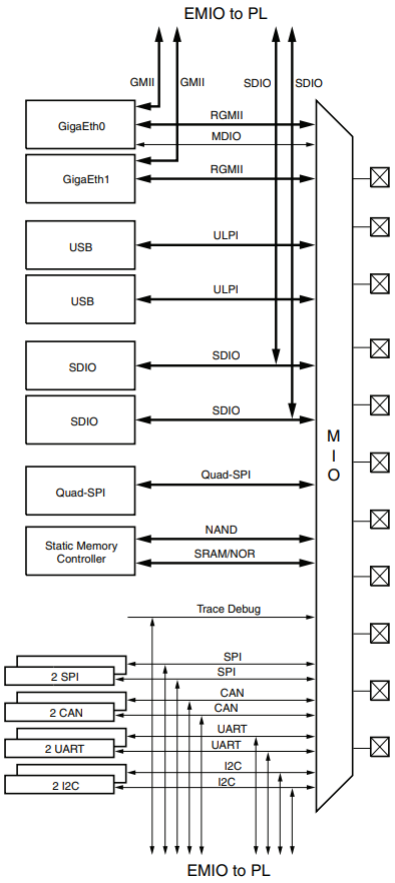


Obr. 1.27: Vnitřní struktura čipu XC7Z010 [13]

1.3.1 Processing system (PS)

Mezi hlavní části [13] PS patří Aplication Processor Unit (APU), paměťová rozhraní (Memory Interfaces), I/O Peripherals (IOP), a propojovací struktura. APU je pro XC7Z010 tvořena již zmíněným dvoujádrovým procesorem ARM Cortex-A9 pracujícím na maximální frekvenci 667 MHz. Tento procesor podporuje výpočty v pohyblivé řádové čárce s jednoduchou i dvojitou přesností (single/double precision). Paměť SRAM o veliksoti 256 KB je přístupná jak z PS tak i z PL. Osmi kanálový DMA, kde čtyři kanály jsou určeny pro PL, umožňuje přenosy z paměti do paměti nebo do periferií a naopak a je možné s ním komunikovat po rozhraní Advanced eXtensible Interface (AXI). Paměťová rozhraní podporují paměti typů např. DDR2, DDR3 (s rychlostí až 1333 Mb/s), NAND a Quad-SPI flash paměti. DDR paměti podporují 16 bitová i 32 bitová slova. Vstupně/výstupní periferie (IOP) zahrnují perieferie, které umožňují datovou komunikaci. Patří sem Gigabitový Ethernet, USB, CAN, SD/SDIO rozhraní pro paměťové karty, UART, I2C nebo SPI. IOP periferie

komunikují s externími zařízeními pomocí 54 dedikovaných Multiplexed I/O (MIO) pinů. Všechny MIO piny podporují 1,8 V, 2,5 V a 3,3 V. Funkce MIO je multiplexovat přístup z periferií PS na piny PS. Pokud jsou třeba dodatečné I/O piny, je možné PS periferie propojit skrze PL na I/O piny, které přísluší PL skrze Extended MIO (EMIO). Strukturu lze vidět na obr. 1.28. Všechny zmíněné části, tzn. APU, paměťová rozhraní a IOP jsou propojena s PL a mezi sebou pomocí ARM AMBA AXI sběrnice. Propojení není blokující a podporuje více současných Master-Slave přenosů.



Obr. 1.28: Struktura MIO [13]

1.3.2 Programable Logic (PL)

Mezi hlavní části PL patří mimo jiné Configurable Logic Block (CLB), Block RAM (BRAM), Digital Signal Processing (DSP) bloky, Programovatelné I/O bloky. Základem CLB je Look-Up Table (LUT). Ta může být konfigurovaná jako 5 nebo 6 vstupová. Každý výstup LUT může být registrový, tzn. může na jeho výstup být připojen klopný obdod typu D. Čtyři takovéto LUT a osm klopných obvodů typu D

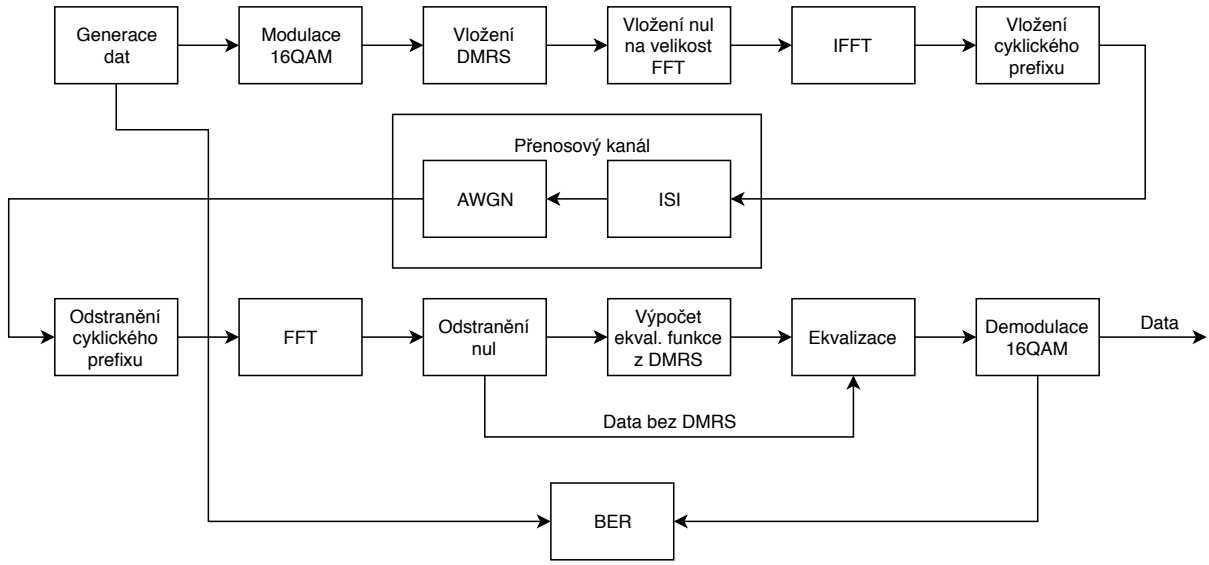
stejně tak jako multiplexery a logika aritmetiky tvoří tzv. slice. Kombinací dvou sliců dostaneme CLB. Navíc čtyři z osmi klopných obvodů typu D v každém slicu mohou být konfigurovány jako latch. Cca 20–50% všech sliců může být nakonfigurováno, aby jejich LUT byly použity jako distribuované RAM paměti nebo posuvné registry.

DSP bloky využívají vysoký počet násobiček a akumulátorů. DSP umožňují násobení binárních čísel až do bitové šířky 18 bitů x 25 bitů ve dvojkovém doplňku s ukládáním do 48 bitového akumulátoru. Násobičky i akumulátory jsou schopny pracovat do frekvence 741 MHz.

Každé I/O je konfigurovatelné na široký počet standardů. I/O porty se dělí na High Range (HR) a na High Performace (HP). HR I/O podporují široký rozsah napětí, od 1,2 V do 3,3 V. HP I/O jsou optimalizovány pro nejvyšší možný výkon pro přenos. Všechny I/O piny jsou organizovány do bank, kde jedna banka obsahuje 50 pinů. Každá banka má vlastní napájení. Každý pin má volitelný pull–up nebo pull-down rezistor. Většina signálních pinů může být konfigurována jako diferenciální s terminačním odporem 100 Ω. Jakýkoliv vstup může být nastaven jako kombinační nebo registrový nebo může být zpožděn o daný čas.

2 ŘEŠENÍ

2.1 Simulace v prostředí MATLAB

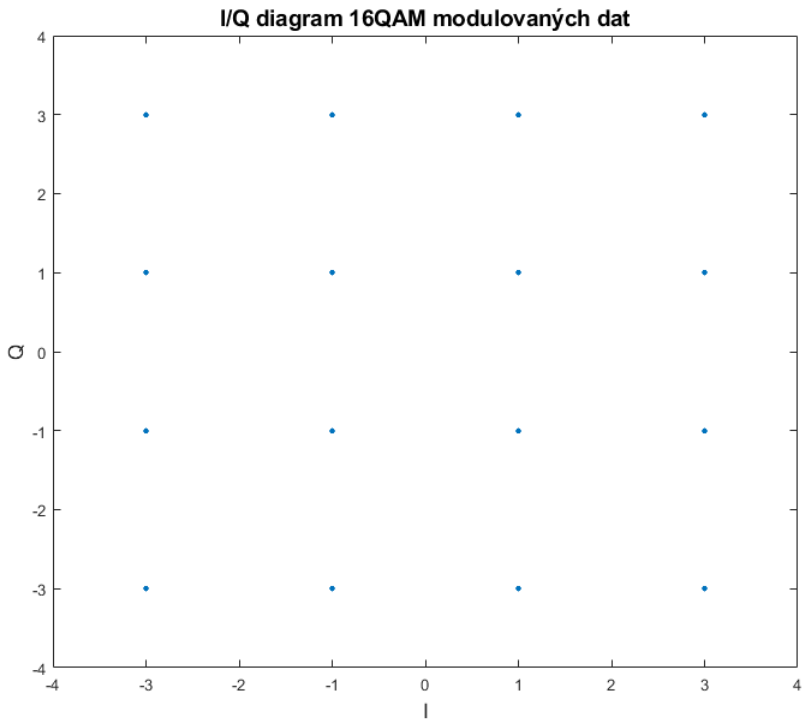


Obr. 2.1: Blokové znázornění toku dat při simulaci v prostředí Matlab

Průběh simulace je znázorněn na obr. 2.1. Před generováním bitové posloupnosti jsou vypočteny parametry pro modulátor pomocí funkce `f_calc_params()`. Ze zadané šířky pásma a vzdálenosti mezi nosnými je vypočítán odpovídající počet datových nosných, délka prvního cyklického prefixu (jeden vždy po 0,5 ms, tzn. v polovině subrámce) a ostatních cyklických prefixů. Funkce také definuje konstanty, např. velikost FFT, která je 4096. Následuje generování dat. Generování dat se provádí funkcí `randi()`. Vygenerovaná data je třeba rozdělit na k bitů na symbol podle vztahu

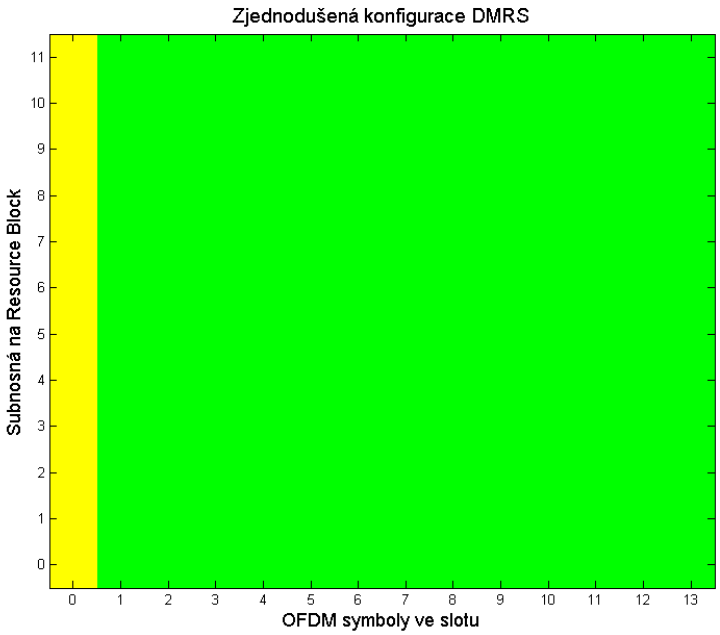
$$k = \log_2 M, \tag{2.1}$$

kde M je řád modulace. Pak jsou data rozdělená po k bitech převedena na dekadická čísla, která se modulují pomocí funkce `qammod()`. V mém konkrétním případě, pro který jsou dále uvedeny výsledky, jsem použil M = 16. I/Q diagram modulovaného signálu je na obr. 2.2.

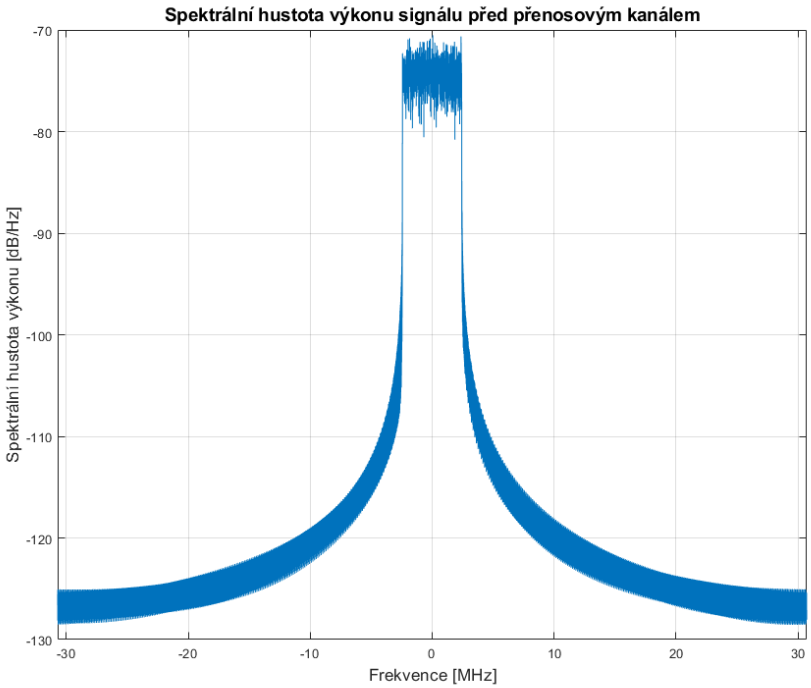


Obr. 2.2: I/Q diagram 16QAM modulovaných dat

Po modulaci následuje vložení DMRS. Symboly jsou nejprve vygenerovány pomocí funkce `randsrc()`, kde počet řádků a sloupců matice s DMRS je dán podle zvolené konfigurace (obr. 2.3). Následně se DMRS přidají k datům. DMRS jsou pro zjednodušení generovány na všech subnosných a zároveň pouze na prvním symbolu. Nicméně toto zjednodušeni nemá vliv na funkčnost vyrovnávání frekvenční charakteristiky kanálu v případě Single Input Single Output (SISO) systému, který ve své práci uvažuji. Jelikož je počet datových nosných vždy menší než celková velikost FFT, DMRS jsou rozděleny na polovinu a mezi ně se vkládá `N_fft - M_data` nul. Po vložení nulových nosných následuje IFFT funkce `ifft()`, do které data vstupují ve formě matice, kde řádky jsou tvořeny jednotlivými subnosnými a sloupce jednotlivými OFDM symboly. Příklad spektrální hustoty výkonu signálu je vidět na obr. 2.4 .



Obr. 2.3: Použitá zjednodušená konfigurace DMRS



Obr. 2.4: Příklad spektrální hustoty výkonu OFDM signálu

Po IFFT následuje vložení cyklického prefixu. Cyklický prefix je vkládán funkcí `f_insert_cp()`. Pro přidání cyklického prefixu potřebuje funkce velikost FFT, délky cyklických prefixů, vzdálenost mezi nosnými a vstupní data ve formě jedno-řádkového vektoru. Výstupem je opět jedno-řádkový vektor.

Následuje simulace přenosového kanálu. Je přidáno vícecestné šíření a AWGN šum. Funkce `conv()` vypočítá konvoluci mezi vektorem dat a impulzní charakteristikou h, která simuluje vícecestné šíření v přenosovém kanálu. Impulzní charakteristiku jsem zvolil ve tvaru `h = [1 zeros(1,31) 0.5 zeros(1,123) 0.1]`. Signál se šíří 3 cestami, kde první cesta přenese signál beze změny. Druhá cesta jej přenese zpožděný o 31 vzorků a zeslabený na polovinu původního signálu. Třetí cesta má zpoždění 123 vzorků a úroveň původního signálu klesne na desetinu. Jedná se o případ, kdy zpoždění způsobené vícecestným šířením nepřesahuje délku cyklického prefixu a tedy ISI by nijak nemělo ovlivnit signál. Vzorkovací perioda je

$$T_{vz} = \frac{1}{\Delta f \cdot N_{fft}} = \frac{1}{15000 \cdot 4096} = 16,28 \text{ ns}. \tag{2.2}$$

Při nejmenší délce cyklického prefixu 288 symbolů (pro $\Delta f = 15$ kHz a $N_{fft} = 4096$) je doba jeho trvání dána

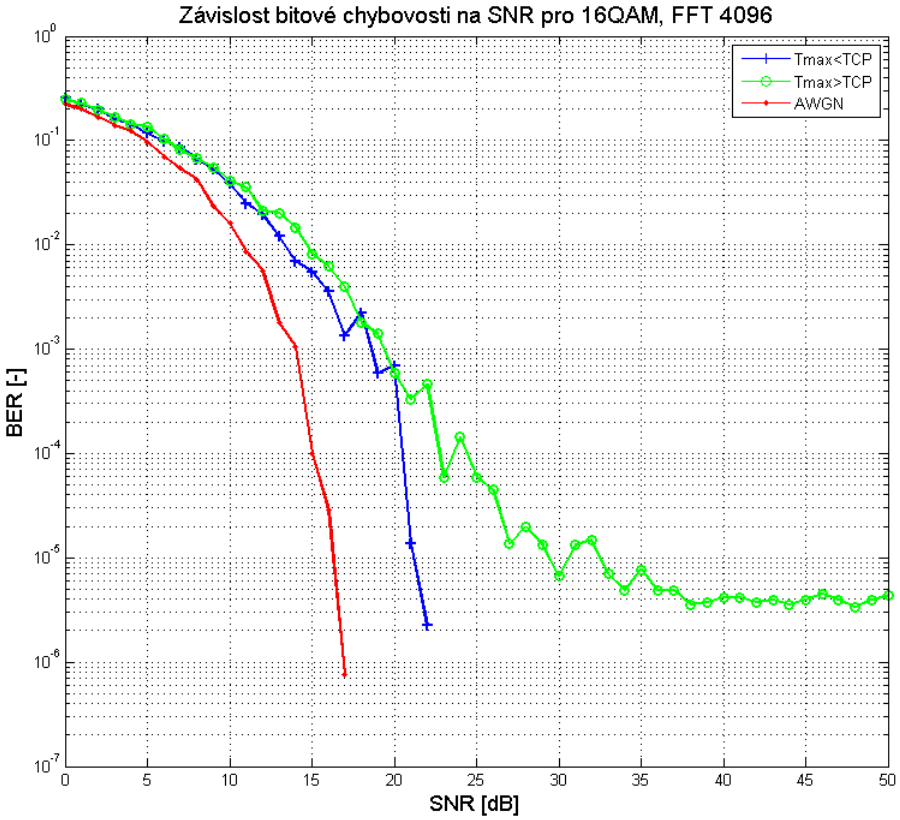
$$T_{cp} = T_{vz} \cdot N_{cp} = 16,28 \cdot 10^{-9} \cdot 288 = 4,69 \text{ µs}. \tag{2.3}$$

Pak maximální zpoždění, kdy nedojde k narušení užitečné informace symbolů z důvodu ISI je $4,69$ µs. Zpoždění jsem zvolil následovně. První cesta nebude mít zpoždění žádné, tedy $\tau_1 = 0$ s. Pro druhou cestu jsem zvolil zpoždění $\tau_2 = 0,5$ µs a pro třetí cestu jsem zvolil zpoždění $\tau_3 = 2$ µs. V počtu vzorků je to pak

$$N_{cp2} = \frac{\tau_2}{T_{vz}} = \frac{0,5 \cdot 10^{-6}}{16,28 \cdot 10^{-9}} \doteq 31 \text{ vzorků} \tag{2.4}$$

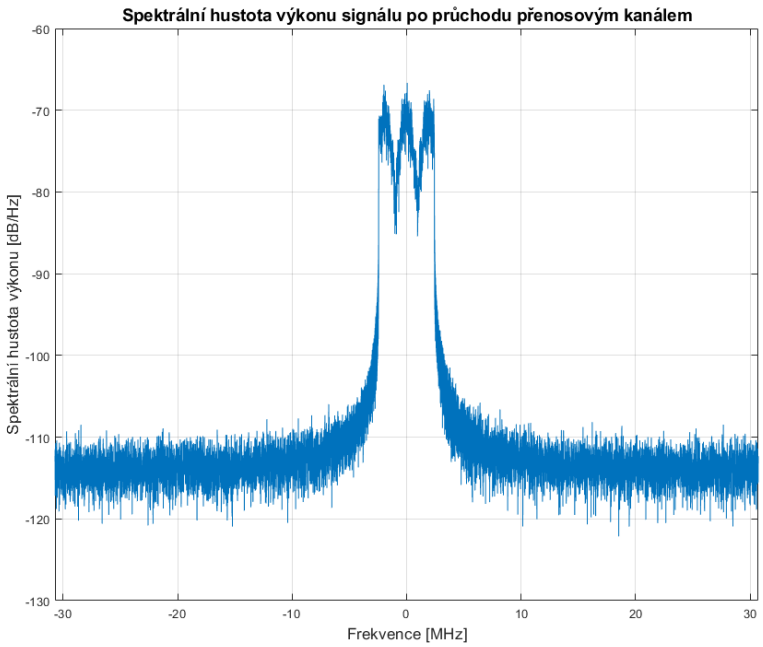
$$N_{cp3} = \frac{\tau_3}{T_{vz}} = \frac{2 \cdot 10^{-6}}{16,28 \cdot 10^{-9}} \doteq 123 \text{ vzorků}. \tag{2.5}$$

Maximální zpoždění by tedy nemělo být delší, než je délka cyklického prefixu. Na obr. 2.5 jsou vidět průběhy bitové chybovosti v závislosti na SNR pro případ, kdy zpoždění v kanálu je menší než délka cyklického prefixu (Tmax<TCP), větší než délka cyklického prefixu (Tmax>TCP). Poslední křivka je pouze pro AWGN kanál bez ISI. Impulzní charakteristika kanálu pro případ (Tmax>TCP) je `h = [1 zeros(1,200) 0.5 zeros(1,600) 0.1]`.

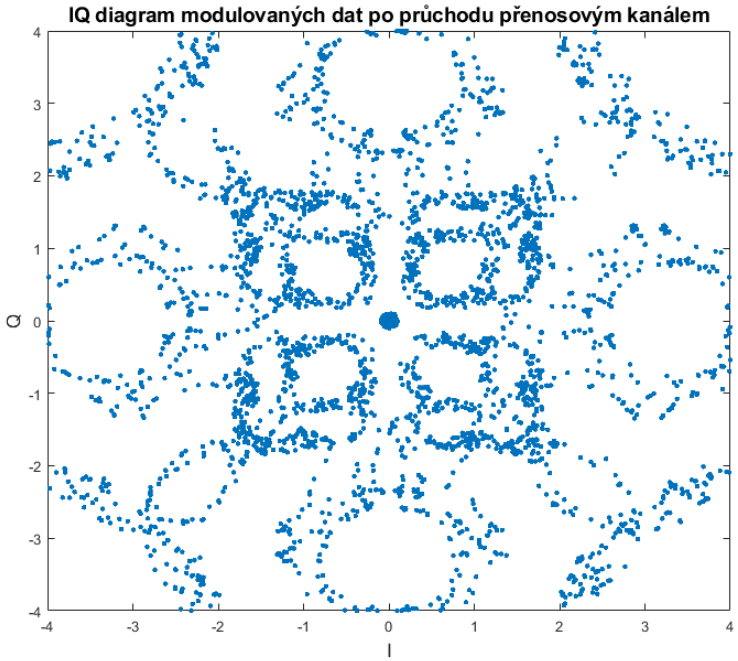


Obr. 2.5: Závislost BER na SNR pro OFDM s 4096 subnosnými, dva modely kanálu, modulace 16QAM, bez kódování

Kromě vícecestného šíření jsou vlastnosti přenosového prostředí simulovány také přidáním aditivního bílého šumu (Additive White Gaussian Noise (AWGN)) s gausovským rozdělením. K tomu se používá funkce `awgn()`, kde SNR je jeden ze vstupních argumentů. Nyní následují předchozí operace, ale v obráceném pořadí. Funkce `f_remove_cp()` zajistí odstranění cyklického prefixu ze vstupního vektoru dat. Na obr. 2.6 je spektrum signálu po průchodu kanálem a po odstranění cyklického prefixu. Následuje funkce `fft()`, do které data vstupují ve formě matice, kde jednotlivé řádky jsou subnosné a sloupce OFDM symboly. Po FFT dostaneme signál, který je vidět na obr. 2.7. Následuje odstranění nulových nosných.



Obr. 2.6: Spektrální hustota výkonu signálu po průchodu kanálem

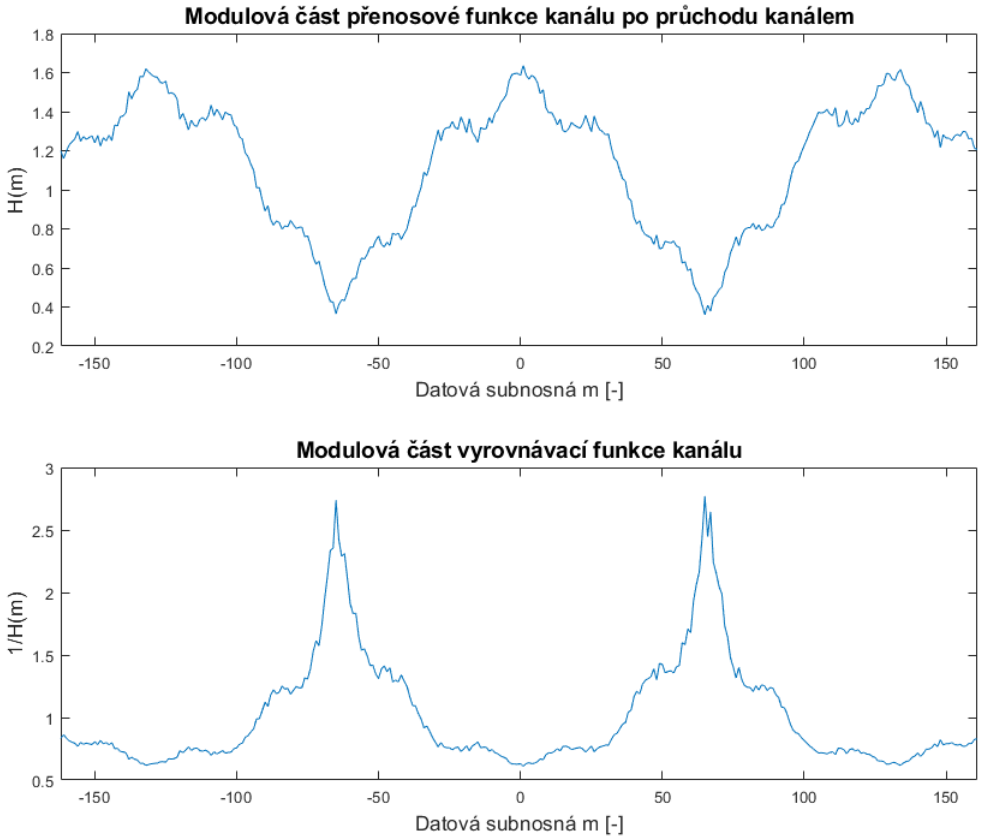


Obr. 2.7: I/Q diagram signálu po průchodu kanálem

Pro vyrovnání přenosové funkce kanálu je třeba nejdříve vypočítat přenosovou funkci kanálu. K tomu slouží DMRS, které jsou mezi daty.
`H = (data_fft_nozeros(:,1))./dmrs_symb` nám zajistí podíl DMRS z přijatého signálu s původními DMRS, které byly přidány do signálu při vysílání. Vyrovnávací funkci pak vypočítáme jako převrácenou hodnotu z přenosové funkce kanálu podle vztahu

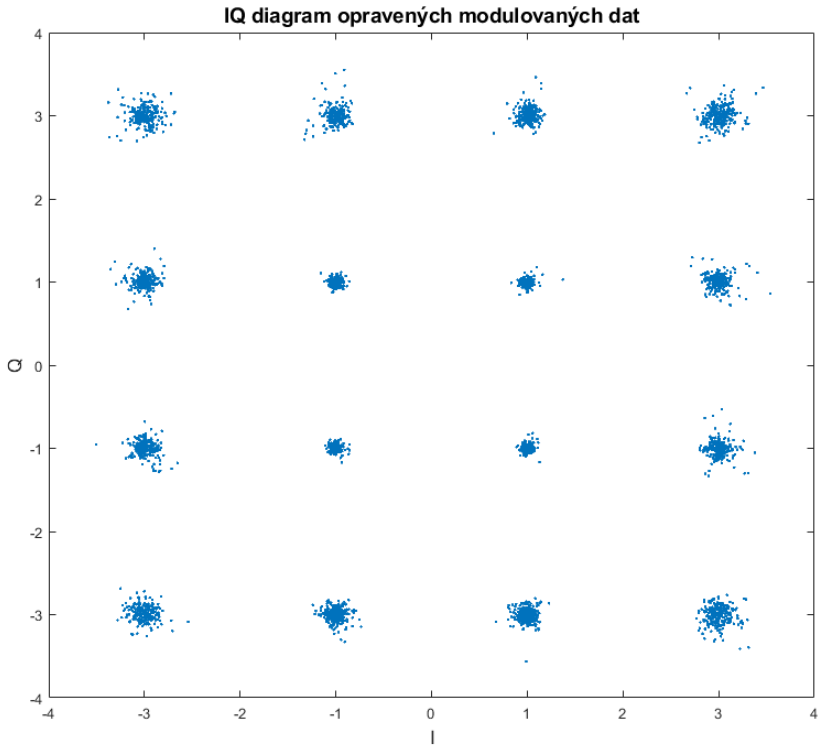
$$Z_m = \frac{1}{H_m} \tag{2.6}$$

kde m = 0, 1, ... M-1. Přenosovou a vyrovnávací funkci kanálu lze vidět na obr. 2.8.



Obr. 2.8: Modulová část přenosové funkce kanálu po průchodu kanálem a vyrovnávací funkce

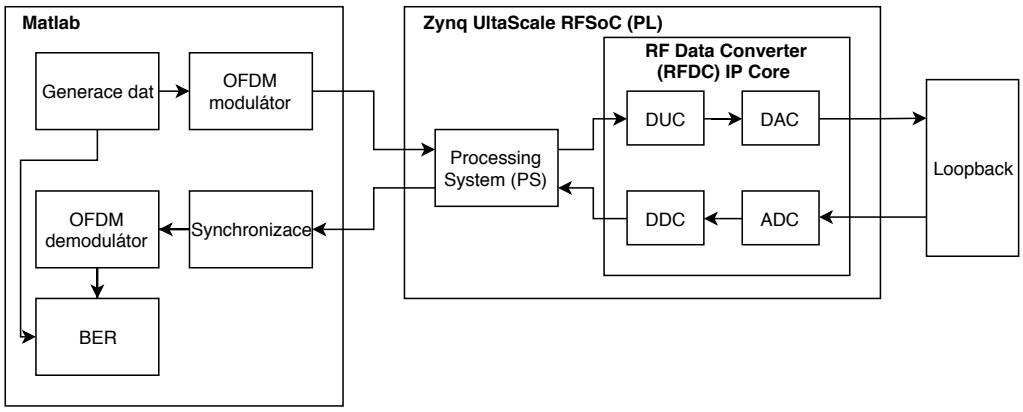
Původní data pak získáme vynásobením všech symbolů přijatých dat s vyrovnávací funkcí. Opravená data v IQ rovině můžeme vidět na obr. 2.9. Nyní už stačí data demodulovat funkcí `qamdemod()`. Jako poslední krok je počítána bitová chybovost funkcí `biterr()`.



Obr. 2.9: I/Q diagram opravených dat

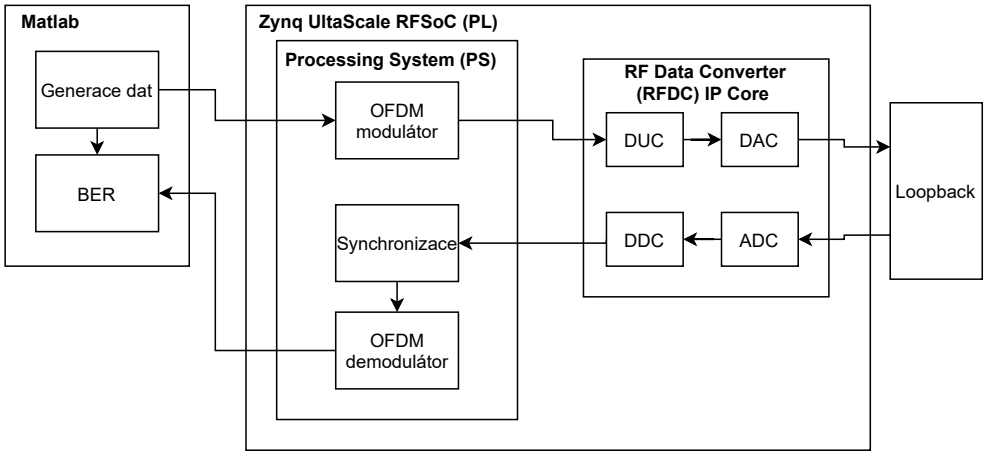
2.2 Koncepce generátoru 5G signálů

Navržená koncepce výsledného generátoru je na obr. 2.10 a obr. 2.11. První varianta spočívá v tom, že Xilinx RFSoC by sloužil pouze pro konverzi signálu z Base Band (BB) na nosnou frekvenci. Celá fyzická vrstva standardu 5G by se vytvářela v simulačním programu, např. Matlab. Tzn. generování nebo načtení dat, následná modulace a další úkony, které jsou v základu shodné s obr. 2.1 v kapitole o simulaci. Výsledná data by byla přenášena z počítače do vývojové desky, odkud by je PS pomocí AXI sběrnice vyslala do RFDC. Ten by zajistil převod z BB na nosnou frekvenci za pomoci jemného mixeru případně DAC (více v kapitole 2.2.1). Pomocí zpětné smyčky (loopback) by byla data přijata, tentokrát přes ADC a převedla by se do BB. Synchronizace by pak probíhala až v simulačním programu.



Obr. 2.10: Blokové schéma koncepce generátoru, první varianta

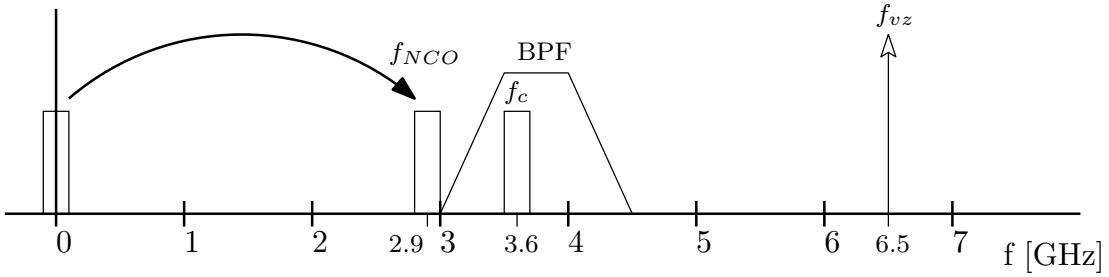
Druhá varianta využívá simulační program pouze pro generování nebo načtení symbolů vnitřní modulace (QAM). Případně by se data mohla posílat přímo do vývojové desky. PS případně PL by vytvářela fyzickou vrstvu, přičemž by se využívalo IP bloků, například pro FFT. Následná konverze na nosný kmitočet a zpět do BB by probíhala stejným principem jako u předchozí varianty. Pro zjednodušení by synchronizace probíhala pro obě varianty na principu korelace přijatého signálu se známým vyslaným signálem. Ve zpětné smyčce (loopback) je třeba u obou variant počítat se zařazením atenuátoru.



Obr. 2.11: Blokové schéma koncepce generátoru, druhá varianta

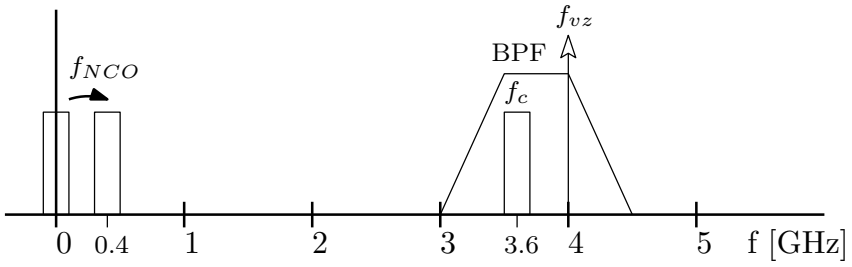
2.2.1 Převod signálu ze základního pásma na nosný kmitočet

Pro převod signálu ze základního pásma (BB) na nosnou frekvenci byly navrženy 3 možnosti. Na obrázcích lze vidět frekvenční spektrum, kde šipka znázorňuje kmitočtový posun signálů pomocí směšovače. Na obr. 2.12 je signál vygenerovaný v základním pásmu. Ten za pomoci jemného směšovače (Fine Mixeru) s NCO převedeme na kmitočet 2,9 GHz. Následným vzorkováním frekvencí $f_{vz} = 6,5$ GHz se objeví kopie signálu na 3,6 GHz a na 9,4 GHz. Pásmovou prospustí s mezními kmitočty 3 GHz a 4,3 GHz odfiltrujeme nechtěné kopie po vzorkování a zůstane nám pouze signál na frekvenci 3,6 GHz. Pro tento případ se pohybujeme ve 2. Nyquistově zóně. Daná Nyquistova zóna ať už pro tento případ nebo pro další dva případy by měla být zohledněna při konfiguraci RFDC.



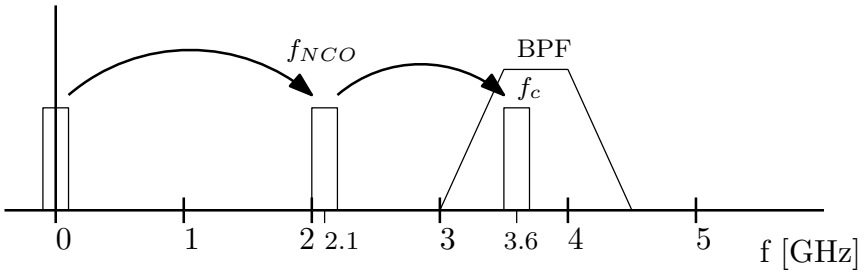
Obr. 2.12: První typ up-konverze

Další způsob lze vidět na obr. 2.13. Signál ze základního pásma převedeme pomocí jemného směšovače s NCO na frekvenci 0,4 GHz. Tento signál pak budeme vzorkovat frekvencí $f_{vz} = 4$ GHz, což nám vytvoří kopie signálu na frekvencích 3,6 GHz a 4,4 GHz. Použitím stejné pásmové propusti jako v předchozím případě vyfiltrujeme všechny nežádoucí kopie kromě té na kmitočtu 3,6 GHz. Pro tento případ se opět nacházíme ve druhé Nyquistově zóně.



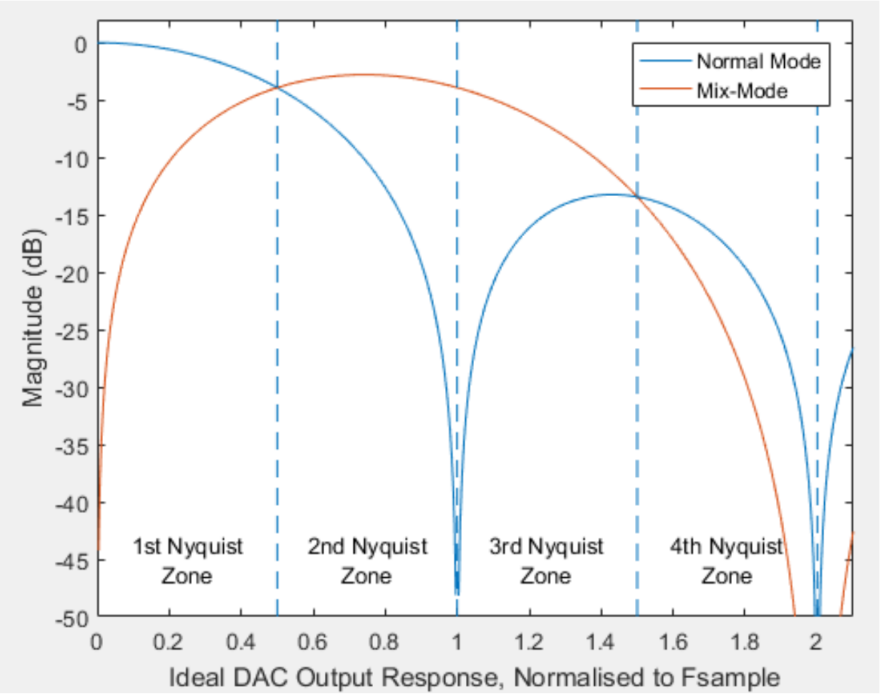
Obr. 2.13: Druhý typ up-konverze

Poslední způsob je na obr. 2.14. Zde bychom využili pouze jemný směšovač, který by signál z BB převedl na 2,1 GHz. Další konverze, například za pomoci frekvence 1,5 GHz na kmitočet 3,6 GHz by provedl externí analogový směšovač. RFDC by tedy prováděl pouze převod na mezilehlou frekvenci $f_{IF} = 2,1$ GHz.



Obr. 2.14: Třetí typ up-konverze

Signál DAC je na výstupu zkreslen frekvenční charakteristikou ve tvaru $\frac{\sin x}{x}$. Jak je vidět na obr. 2.15, pro signály blízko vzorkovací frekvence nastává výrazný pokles amplitudy (modrý průběh). DAC má pomocí FIR filtru implementovanou kompenzaci funkce sinc a tedy zajistí vyrovnání výstupní charakteristiky. Obvod umožňuje kompenzovat pokles výkonu na výstupu DAC ve vyšších Nyquistových zónách. Výrobce definuje tzv. Mixed mode, viz. obr. 2.15. Při správně zvolené Nyquistově zóně, ve které chceme pracovat, můžeme dosáhnout výrazného zvýšení výkonu výstupního signálu právě pro signály blízko vzorkovací frekvence (oranžová křivka).



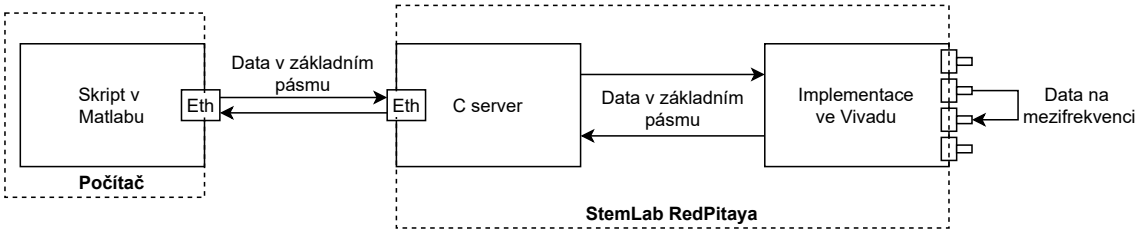
Obr. 2.15: Operace DAC v Nyquistových zónách [10, RFDC LogiCORE IP]

2.2.2 Konfigurace RFDC

V tab. A.2 resp. A.1, které jsou umístěny v příloze této práce, je uveden souhrn parametrů pro nastavení RFDC a DAC resp. ADC pro varianty popsané v kapitole 2.2.1. Důležité je nastavit povolení inverzního sinc filtru a správnou Nyquistovu zónu. Dále také povolit jemný směšovač (Fine Mixer) a frekvenci NCO spolu se vzorkovací frekvencí převodníku.

2.3 Implementace generátoru

Pro implementaci generátoru byl ve druhé části práce zvolen vývojový kit STEMLab Red Pitaya 125-14. Původně plánovaný kit Zynq UltraScale+ RFSoC ZCU111 nemohl být z důvodu jeho pozdního dodání dodavatelem použit. Při vytváření výsledného generátoru jsem vycházel převážně z [19] a z materiálů předmětu Implementace softwarových komunikačních systémů (MIKS) [29]. Předmět MIKS je vyučován na Fakultě Elektrotechniky a Komunikačních Technologií (FEKT). Některá z IP jader použitých v IP Integrátoru programu Vivado jsou převzata z [19].



Obr. 2.16: Blokové schéma výsledného generátoru

Jak je vidět na obr. 2.16, implementace generátoru sestává ze tří hlavních částí. První část je tvořena skriptem v programu Matlab. Hlavní část tohoto skriptu tvoří simulační skript 5G OFDM signálu, který je popsán v kapitole 2.1. Zajišťuje generování dat určených k vysílání v základním pásmu, ekvalizaci a vyhodnocení přijatých dat. Do toho skriptu byla přidána funkce pro odesílání a příjem dat do/z C serveru a následná časová synchronizace přijatého signálu. Podrobnějším popisem se zabývá kapitola 2.3.1. Další částí je server napsaný v jazyku C. Server zajišťuje spojení mezi Matlabem a designem ve Vivadu. Pracuje jako Transmission Control Protocol (TCP) server, který zajišťuje přijímání i odesílání dat z/do Matlabu. Tyto data kopíruje do/z vyrovnávacích pamětí, odkud/kam se kopírují do/z paměti, která přísluší portu GP0/HP0. Podrobnějším popisem se zabývá kapitola 2.3.2. Třetí část je tvořena designem, který je implementován do PL Field Programmable Gate Array (FPGA) čipu. Tento design pracuje s daty, která jsou poskytnuta C serverem. Jsou to data komplexní (soufázová složka I a kvadraturní složka Q), která jsou v designu interpolována, namodulována na nosný kmitočet a převedena pomocí DAC na analogový signál. Smyčkou se tento signál vrací na ADC a takto zdigitalizovaná data jsou převedena do základního pásma, decimována a připravena pro vyčtení C serverem. Podrobnějším popisem se zabývá kapitola 2.3.3.

2.3.1 Matlab

Jak již bylo zmíněno, podstatná část kódu vychází ze simulace OFDM signálu, kterou se zabývá kapitola 2.1. V části skriptu, která náleží přenosovému kanálu bylo nutné vytvořit funkci, která bude zajišťovat odeslání a příjem dat. Následně bylo také třeba přidat část kódu, který bude zajišťovat časovou synchronizaci signálu včetně úpravy fáze mezi signálem vyslaným a signálem přijatým.

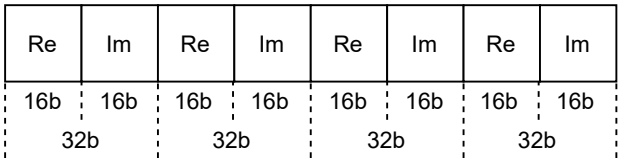
Funkce, která zajišťuje odesílání a příjem dat (tedy komunikaci s C serverem) se nazývá `f_rp_tran_acq()`. Na výpisu 2.1 je popis vstupních a výstupních parametrů funkce. Za vstupní argument `host` se vkládá IP adresa RP.

Výpis 2.1: Argumenty funkce pro komunikaci se serverem

```
function [data_Re, data_Im, data_all_Re, data_all_Im] ...
    = f_rp_tran_acq( ...
        data_in_Re, data_in_Im, cmplx_data, data_in_samples, ...
        out_width, txrx_no, bytes_to_read, thld, host, port)
%   Detailed explanation goes here
    %Inputs:
    % - data_in_Re = Real part of input data
    % - data_in_Im = Imaginary part of input data
    % - cmplx_data = if 1, then Real and Complex part of number ...
       must be passed as input argument, data are then interleaved
    % - out_width = Width of data to be send over TCP in number ...
       of bits included sign
    % - txrx_no = Number of transmit and receive loops
    % - bytes_to_read = Number of Bytes to read from TCP server
    % - thld = Treshold Voltage received data
    % - host = IP address of TCP host
    % - port = TCP port of host
    %Outputs:
    % - data_Re = Real part of valid received data
    % - data_Im = Imaginary part of received valid data
    % - data_all_Re = Real part of all received data
    % - data_all_Im = Imaginary part of all received data
```

Zdrojový kód funkce lze nalézt v přílohách pod názvem `f_rp_tran_acq.m`. Algoritmus (inspirovaný z [29]) funkce je následující. Data určená k vyslání (reálná a imaginární složka) jsou nejdříve převedena z formátu double na 16 bitový znaménkový integer. Pokud je zvoleno, že nebudeme pracovat s komplexními čísly (`cmplx_data = 0`), do proměnné imaginární složky se uloží nulová hodnota, která se bude předávat až na výstup. Do předem alokované proměnné se následně uloží 16 bitová integer čísla tak, aby se střídala reálná a imaginární složka. Vznikne tedy

vektor, který je znázorněn na obr. 2.17. Tímto získáme komplexní vzorek s šířkou 32 bitů, kde reálnou část tvoří 16 bitů a imaginární část také 16 bitů. Tento vektor je nutné převést pomocí funkce `typecast()` na neznaménkový 8 bitový integer, který je možné následně převést na textový řetězec. Textový řetězec je určen k odeslání přes TCP. Pomocí Matlab Executable (MEX) funkce `pnet()` (převzata z [29]) je navázáno TCP spojení se serverem, který běží na RP. Po úspěšném připojení se po dobu `txrx_no` cyklů vykonává smyčka `for-loop`. Ve smyčce dochází k odeslání a následnému příjmu dat. Pokaždé, co jsou data přijata, dochází k jejich konverzi a uložení. To znamená, že textový řetězec je převeden na hodnoty jednotlivých znaků pomocí modulo dělení a následně přetypován na neznaménkový 8 bitový integer. Tento vektor je pak pomocí funkce `typecast()` převeden na vektor 16 bitových znaménkových čísel. Každý komplexní vzorek je pak tvořen dvojicí 16 bitových čísel (reálnou a imaginární složkou). Vektor reálných a imaginárních složek je převeden na typ double. Tyto double data jsou v každé smyčce ukládána do vektoru. Po skončení smyčky se v těchto datech detekuje předem zvolený práh (`thld`) a od něj se vybere blok dat o velikosti rovné vyslaným datům. Na výstup funkce `f_rp_tran_acq()` se přiřazuje vektor obsahující veškerá data ze všech opakování `for-loop` smyčky a také pouze blok dat detekovaný zvoleným prahem (`thld`).



Obr. 2.17: Způsob prokládání dat

Základem časové synchronizace je funkce `f_time_sync()`, jejíž princip v subsamplové synchronizaci vychází z [28]. Funkce se skládá ze dvou hlavních částí, ze synchronizace celočíselné a frakční (subsamplové). Při celočíselné synchronizaci je počítána korelace funkcí `xcorr()` mezi signálem přijatým a signálem vyslaným. Funkce `xcorr()` vrací korelaci a vektor zpoždění pro danou korelaci. Pro pozici maximální korelace je zjištěna hodnota zpoždění. O tuto hodnotu zpoždění je přijatý signál posunut pomocí funkce `circshift()`.

Subsamplová synchronizace probíhá ve smyčce, pomocí které je přijatý signál zpožďován v rámci vzorkovací periody T_{vz}. Vektor zpoždění je definován od $-0.5T_{vz}$ do $0.5T_{vz}$ s krokem 0.02. Pro každé zpoždění je vypočítána korelace mezi zpožděným přijatým signálem a signálem vyslaným. Výsledky jsou následně ukládány. Po ukončení cyklu `for-loop` je nelezeno maximum z uložených výsledků a pro toto maximum se přijatý signál subsamplově posune. Pro tento způsob časové synchronizace

se předpokládá, že na přijímači známe kromě přijatého signálu také signál vyslaný. To je ale dostačující, neboť návrh synchronizátoru není součástí práce.

Po časové synchronizaci je volána funkce `lscov(A, B)` pro fázové dotočení přijatého komplexního signálu. Funkce vrací vektor komplexních hodnot, kterými je třeba vynásobit přijatý signál, abychom dostali signál co možná nejvíce podobný původnímu, vyslanému signálu. Funkce je založena na řešení lineární soustavy rovnic $A * x = B$ pomocí metody nejmenších čtverců.

2.3.2 C server

Jak již bylo řečeno v kapitole 2.3, server `adc_dac_server.c` je prostředek pro komunikaci mezi uživatelem a RP, přes který je možné posílat i přijímat data. Server je z velké části převzat z projektů [19] a jeho popis je uveden především pro kompletnost celého popisu.

Abychom mohli přistupovat do paměti, která přísluší FPGA čipu, je třeba namapovat adresní prostor podle editoru adres z Vivada (obr. 2.18) pomocí funkce `mmap()`. Díky editoru adres víme, jaké úseky paměti používáme, jejich adresy a velikost. Adresní prostor na FPGA čipu Zynq XC7Z010 je definován v souboru `/dev/mem` na RP. Otevřením souboru pro čtení i zápis je získán identifikátor souboru, který slouží jako vstupní argument pro funkci mapování `mmap()`. Takto je třeba namapovat konfigurační registr, stavový registr a Zapisovač, jež jsou připojeny k portu GP0 a jsou umístěny v paměti PL ([14] str. 112). Namapovat je také třeba Double-Data Rate (DDR) paměť portu HP0 ([14] str. 112).

Diagram | Address Editor

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
ps_0					
Data (32 address bits : 0x40000000 [1G])					
cfg_0	S_AXI	reg0	0x4000_0000	4K	0x4000_0FFF
sts_0	S_AXI	reg0	0x4000_1000	4K	0x4000_1FFF
writer_1	S_AXI	reg0	0x4001_0000	64K	0x4001_FFFF
writer_0					
M_AXI (32 address bits : 4G)					
ps_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF

Obr. 2.18: Editor adres ve Vivadu

Jelikož server běží pod operačním systémem Ubuntu Linux, který má paměť dělenou po tzv. stránkách (pages), je nutné při práci s pamětí pracovat s násobky těchto stránek. Velikost stránky je definována jako 4096 Bytů. Pro paměť, pomocí

níž posíláme data do DAC, je zvolena velikost 16*4096 Bytů. To umožňuje posílat 32768 vzorků po 16 bitech nebo 16384 vzorků po 32 bitech. Velikost paměti pro příjem dat z ADC je zvolena na 2048*4096 Bytů.

Po namapování adresního prostoru probíhá resetování bloků. K tomu je používána syntaxe:

```
*((uint16_t *)(cfg + 0)) |= 2
```

To znamená, že chceme zapisovat dekadickou hodnotu 2 na adresu (ve formátu `uint16_t`) konfiguračního registru zvýšenou o 0 Bytů. Dojde tedy k nastavení druhého nejnižšího bitu v nejnižším Bytu konfiguračního registru. V proměnné `cfg` máme díky předchozímu mapování uloženou počáteční adresu konfiguračního registru.

Hlavní program je rozdělen pomocí `fork()` na dva procesy, rodičovský a dceřiný proces. Dceřiný proces je používán pro komunikaci mezi PS a serverem a obráceně. Zajišťuje tedy kopírování navzorkovaných dat z ADC do vyrovnávací paměti `buf_adc` a pak také kopírování dat připravených pro DAC z vyrovnávací paměti `buf_dac`. Rodičovský proces pak inicializuje TCP komunikaci a odesílá navzorkovaná data z vyrovnávací paměti `buf_adc` připojenému klientovi přes TCP protokol. Následně pak také od připojeného klienta přijímá data, která jsou určena k odeslání do DAC.

Jelikož se rodičovský a dceřiný proces vykonávají současně, je k řízení přístupu do vyrovnávacích pamětí `buf_adc` a `buf_dac` použito rour (pipes). Ty lze vytvořit voláním funkce `pipe()` a jako vstupní parametr je třeba vložit pole typu integer o dvou pozicích, které bude reprezentovat danou rouru. Pro směr dat do DAC je princip následující: Voláním `close(pipefd_dac[1])` v dceřiném procesu se roura `pipefd_dac` uzavře pro zápis, jelikož z ní budeme chtít později v dceřiném procesu číst. Čtení v dceřiném procesu provedeme voláním

`read(pipefd_dac[0], &buffer_dac, sizeof(buffer_dac))`

a následně zkopírujeme skutečná data do vyrovnávací paměti pomocí funkce

`memcpy(dac, buf_dac, 16*4096)`.

Roury v tomto případě tedy neslouží k přímému předávání dat, ale jsou pouze využívány k řízení přístupu k vyrovnávací paměti, v tomto případě k vyrovnávací paměti `buf_dac`.

V rodičovském procesu je roura `pipefd_dac` uzavřena pro čtení voláním funkce `close(pipefd_dac[0])`, protože budeme chtít zapisovat data, která přijímáme od klienta přes TCP. Po skutečném zápisu dat do vyrovnávací paměti `buf_dac` rouru uvolníme voláním funkce

`write(pipefd_dac[1], &buffer_dac, sizeof(buffer_dac))`.

Tímto způsobem by mělo tedy být zabezpečeno, že se nebude do vyrovnávací paměti `buf_dac` přistupovat ze dvou míst současně. Pro opačný směr dat (navzorkovaná data z ADC) je princip analogický.

V rodičovském procesu pak také probíhá inicializace TCP. Tato inicializace sestává z vytvoření soketu voláním funkce `socket()`. Jako vstupní argumenty je třeba vložit typ IP adresy a typ soketu (zda bude TCP nebo UDP). Funkce `setsockopt()` slouží k nastavení soketu. K soketu je přiřazena IP adresa definovaná strukturou `addr`, kam je třeba vložit typ IP adresy a samotnou IP adresu a port, na kterém bude komunikace probíhat. Adresa je pak přiřazena k soketu funkcí `bind()`. Funkce `accept()` čeká na připojení klienta a dokud k připojení nedojde, blokuje vykonávání programu. Po připojení klienta je navázána komunikace a je tak možné komunikovat oběma směry, tedy server-klient i klient-server. K tomu jsou použity funkce `recv()` a `send()`. Jako vstupní argumenty je třeba vložit identifikátor soketu, kam se mají data uložit/odkud číst a jaká je velikost těchto dat v Bytech. Soket klienta serveru je pak uzavřen voláním funkce `close()`, kde za vstupní parametr je nutné doplnit identifikátor soketu, který chceme uzavřít.

2.3.3 Implementace ve Vivadu

Na obr. B.1 lze vidět blokové schéma designu vytvořeného v IP Integratoru programu Vivado firmy Xilinx. Design vychází z projektu [20]. Základem designu je PS, který přijímá data na portu HP0 (High Performance port (HP)) a odesílá data na portu GP0 (General Purpose port (GP)). Přes port GP0 se tedy odesílají data směrem k DAC. K portu GP0 je také připojen stavový (status) a konfigurační (cfg) registr. Konfigurační registr se používá k resetování bloků jako je převodník bitové šířky a zapisovač Z1. Stavový registr umožňuje uživateli sledovat, z jaké části jsou zaplněné First-In First-Out (FIFO) a zapisovač Z1.

Šířka dat přicházejících z PS na GP port je 32 bitů. Přičemž spodních 16 bitů reprezentuje reálnou a horních 16 bitů reprezentuje imaginární složku komplexního čísla generovaného Matlabem. Toto 32 bitové číslo prochází přes vnitřní propojovací strukturu FPGA čipu do zapisovače Z2. Zapisovač Z2 zajišťuje zápis příchozích dat do FIFO paměti. Z FIFO paměti data vstupují do rozdělovače datového toku R2, který má za úkol oddělit reálnou a imaginární složku komplexního vzorku do dvou 16 bitových paralelních datových toků. Pro změnu vzorkovacího kmitočtu jsou použity FIR interpolátory I1 a I2, které zvyšují vzorkovací frekvenci signálu z 6.25 MSa/s na 125 MSa/s, aby bylo možné signál namodulovat na nosný kmitočet. Výstupy z filtrů se násobí s harmonickým signálem v násobičkách N3 a N4. Harmonické signály v obou větvích jsou vůči sobě posunuty o 90° a jsou generovány blokem Direct Digital Synthesizers (DDS). Vynásobené signály jsou sečteny ve sčí-

tačce a z výsledku je vybráno pouze nejvyšších 16 bitů. Takovéto zapojení násobiček N3, N4, DDS a sčítačky tvoří kvadraturní modulátor. Kvadraturním modulátorem dosáhneme převodu komplexního signálu ze základního pásma na reálný signál na nosném kmitočtu. DAC registr zajistí správné oříznutí dat pro 14 bitový DAC převodník. Výstup DAC0 (na vývojovém kitu RP označeno jako OUT1) je propojen se vstupem ADC1 (na vývojovém kitu RP označeno IN2).

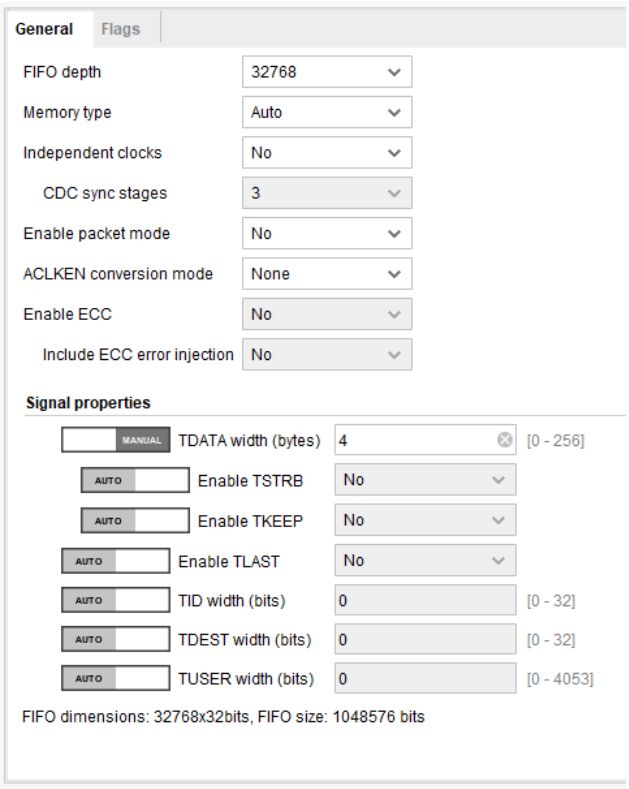
Analogová data z DAC převodníku se smyčkou vrací zpět a vstupují do ADC. Převodník dat vybírá pouze jeden kanál ze dvou dostupných kanálů ADC. Navzorkovaný signál z vybraného kanálu je rozdělen do dvou identických paralelních větví. V každé větvi je tento signál násoben vzájemně fázově posunutým harmonickým signálem z DDS. V tomto případě tvoří násobičky N1, N2 a DDS kvadraturní demodulátor, který navzorkovaný signál posune z nosného kmitočtu do základního pásma. Takto převedená data v základním pásmu se decimují za pomoci FIR decimátorů D1 a D2 ze vzorkovací frekvence 125 MSa/s na 6.25 MSa/s. Návrh FIR filtru je popsán níže. Slučovač datového toku sloučí paralelní 16 bitové datové toky do jednoho datového toku o šířce 32 bitů. Reálná složka opět zaujímá spodní část a imaginární složka horní část 32 bitového čísla. Jelikož je sběrnice HP0 nakonfigurována pro 64 bitů široká data, je použit převodník bitové šířky. Zapisovač Z1 zajistí zápis dat do paměti, která přísluší sběrnici HP0.

Konfigurace bloků ve směru od portu GP0 je následující:

Zapisovač Z2 (AXI4-Stream Writer) [19]

- Šířka adresy v bitech (AXI ADDR WIDTH): 16 bitů
- Šířka dat v bitech (AXI DATA WIDTH): 32 bitů

FIFO paměť (AXI4-Stream Data FIFO)



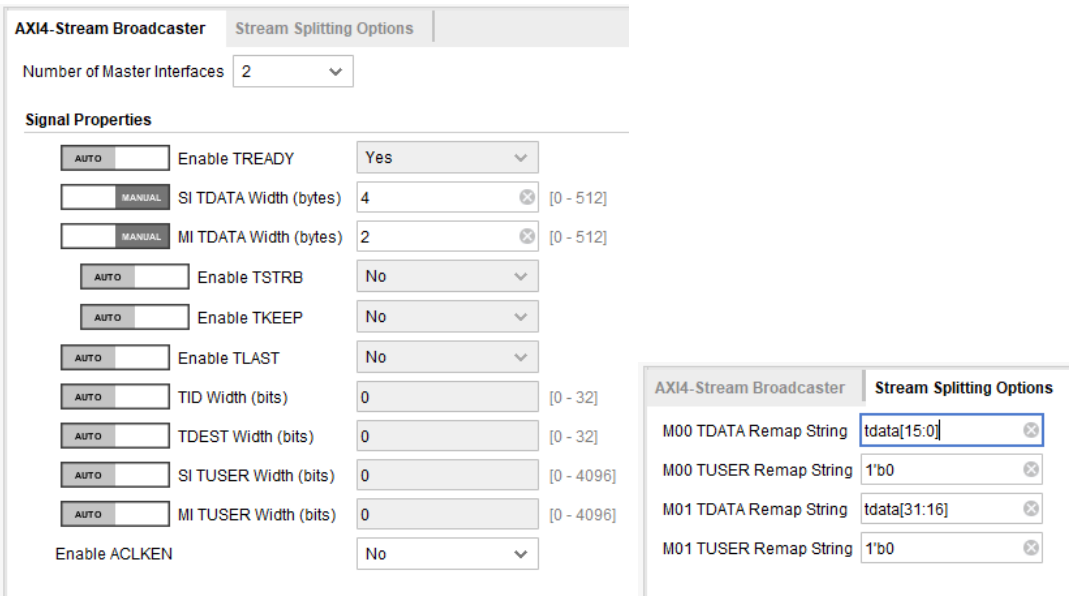
Obr. 2.19: Konfigurace FIFO paměti

Hloubka paměti FIFO byla zvolena o něco větší, než je 16384 vzorků, aby bezpečně obsáhla všechny vzorky, které se posílají z Matlabu a jsou určeny pro vyslání přes DAC. Paměť byla nastavena podle [21].

Nulovač(AXI4-Stream Zeroer) [19]

Zeroer slouží pro nulování svého výstupu, pokud řídící signály AXI4-Stream sběrnice na jeho vstupu signalizují, že nejsou dostupná nová data.

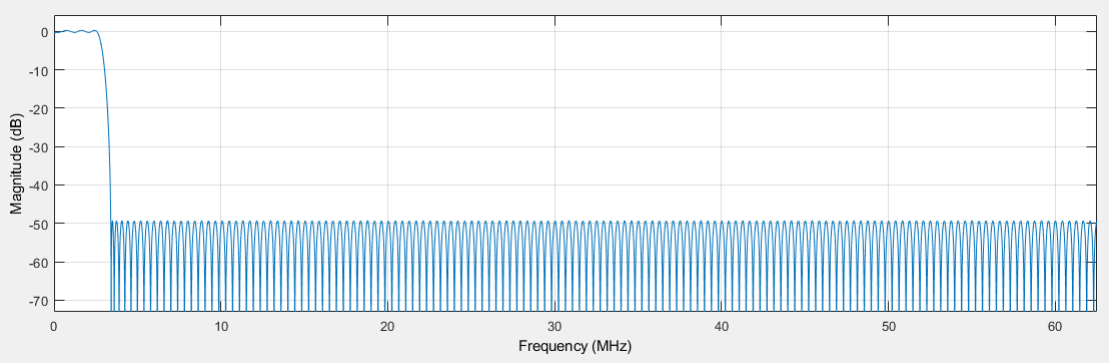
Rozdělovač datového toku R2 (AXI4-Stream Broadcaster)



Obr. 2.20: Konfigurace Rozdělovače datového toku

Jak již bylo zmíněno, je třeba 32 bitové číslo, které reprezentuje jeden komplexní vzorek rozdělit na reálnou a imaginární složku po 16 bitech. Vstup Rozdělovače datového toku je proto nastaven na šířku 32 bitů a výstup (platí pro každý výstup) na 16 bitů dle [22]. Pro správné rozdělení reálné a imaginární části do dvou paralelních větví je nutné přemapovat vstupní data jak je znázorněno na obr. 2.20.

FIR interpolátor I1 a I2 (FIR Compiler)



Obr. 2.21: Modulová charakteristika navržené dolní propusti pro FIR filtr

Pro interpolátory I1 a I2 bylo nejdříve nutné navrhnout filtr typu dolní propust. Filtr byl navržen v nástroji filterDesigner [24] v programu Matlab. Modulovou charakteristiku lze vidět na obr. 2.21. Filtr je navržen vůči výsledné vzorkovací frekvenci $f_s = 125$ MHz. Jako kompromis mezi požadovanou šířkou pásma, poměrně vysokým vzorkovacím kmitočtem a implementační náročností, bylo propustné pásmo nastaveno na $f_p = 2.6$ MHz, zádržné pásmo pak začíná od kmitočtu $f_s = 3.4$ MHz. Zvlnění v propustném pásmu je nastaveno na $A_p = 0.5$ dB a útlum v zádržném pásmu na $A_s = 50$ dB. S tímto nastavením vychází řád filtru na 300. Vypočítané koeficienty jsou kopírovány do FIR Compileru v programu Vivado.

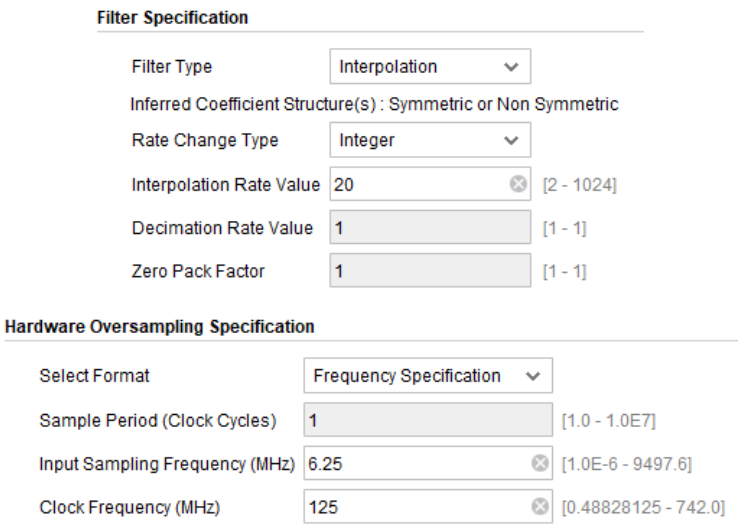
Při takto nastavených filtrech je obsazenost DSP části v FPGA 54%. Strmější filtry by vedly na zvýšení počtu koeficientů a vyšší obsazenost zdrojů v FPGA, v extrémním případě i nemožnost implementace do daného obvodu. Řešením by pak mohlo být sdílení jednoho filtru pro více signálů, případně filtrace pomocí kaskády více FIR filtrů.

Na obr. 2.22 a 2.23, je vidět zvolená konfigurace FIR filtru pro funkci interpolátoru, která byla nastavena podle [23]. Typ filtru je třeba nastavit na interpolační s celočíselným faktorem 20. Interpolační faktor lze získat ze vzahu

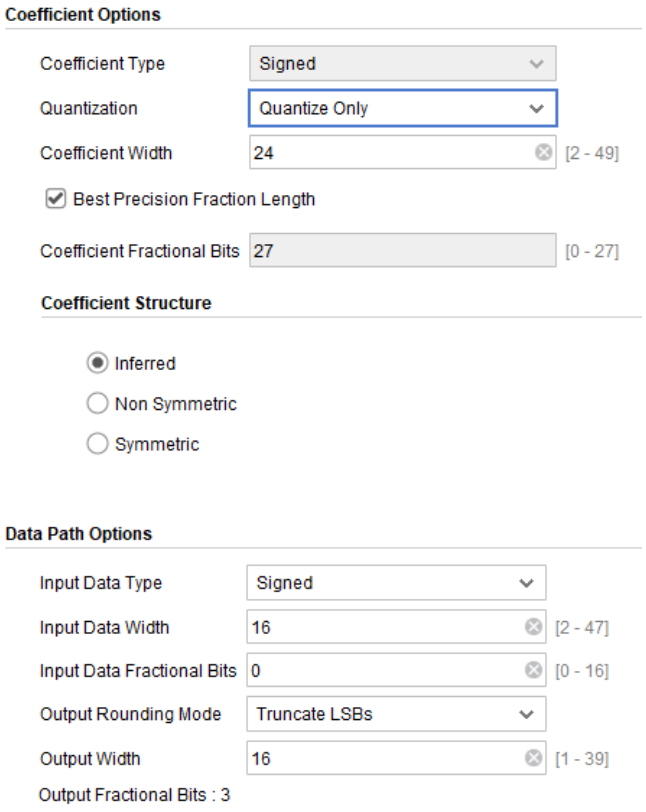
$$L = \frac{f_{výst}}{f_{vst}} = \frac{125 \cdot 10^6}{6.25 \cdot 10^6} = 20 \tag{2.7}$$

kde $f_{výst}$ je kmitočet po interpolaci a f_{vst} je kmitočet před interpolací.

Dále je třeba nastavit jak se mění vzorkovací kmitočet, tedy vstupní vzorkovací kmitočet je 6.25 MHz při systémovém hodinovém signálu 125 MHz. Poslední nastavení se týká kvantizace koeficientů z typu double na typ fixed-point a pak také volba šířky vstupních a výstupních dat. Po vložení v Matlabu vypočítaných koeficientů FIR kompilátor sám zvolí možnost kvantizace Quantize Only. Šířka kvantovaných koeficientů je nastavena na 24 bitů. Při volbě Best Precision Fractional Length FIR kompilátor sám zvolí potřebnou velikost zlomkové části pro nejlepší přesnost. Vstupní data jsou 16 bitová znaménková bez zlomkové části a na výstupu požadujeme 16 bitů. Je dobré myslet na to, že výstup obsahuje 3 zlomkové bity a zohlednit to pak v případě potřeby při vyhodnocování dat.

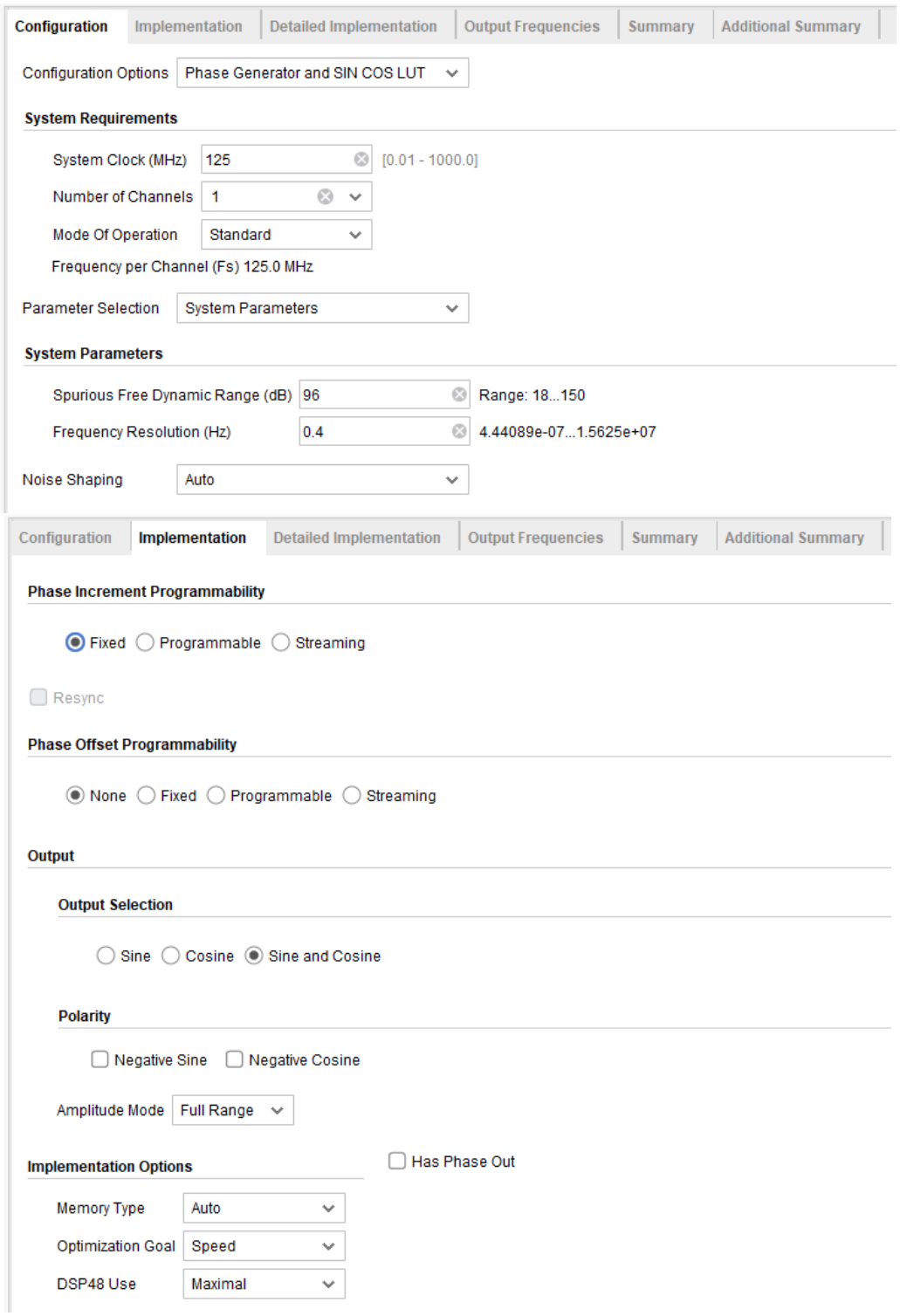


Obr. 2.22: Konfigurace FIR kompilátoru



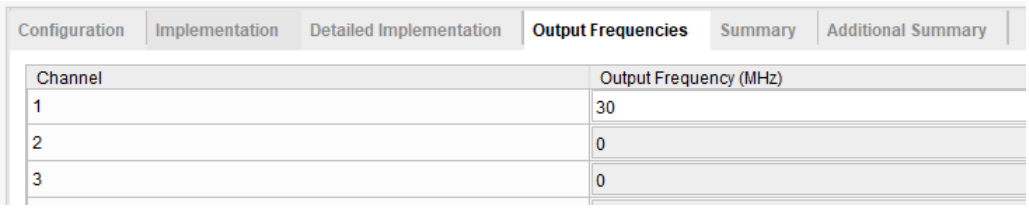
Obr. 2.23: Konfigurace dat FIR kompilátoru

DDS (DDS Compiler)



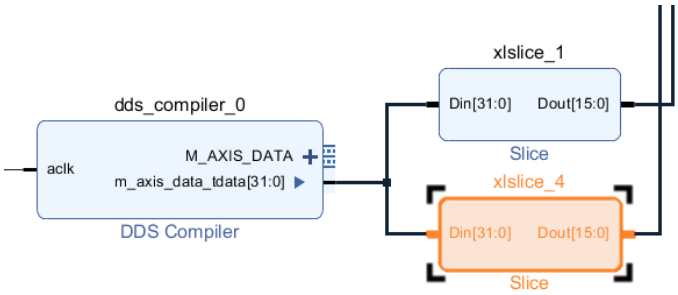
Obr. 2.24: Konfigurace DDS

DDS slouží ke generování harmonických průběhů, které jsou vzájemně posunuty o 90° (sinus a cosinus). Při konfiguraci (obr. 2.24) podle [25] je třeba nastavit systémový hodinový signál na 125 MHz. Bitová šířka dílčích harmonických signálů na výstupu je přímo ovlivňována zvoleným Spurious Free Dynamic Range (SFDR). Hodnota SFDR byla zvolena, aby měly harmonické signály na výstupu 16 bitů. Data na výstupu DDS jsou ve formátu fix16_15 v podobě 32 bitového slova. Formát fixM_N znamená, že číslo je uloženo ve formátu fixed-point o celkovém počtu M bitů, z nichž N bitů je zlomových. Spodních 16 bitů výstupu DDS (bity 15:0) obsahuje signál cosinus, horních 16 bitů výstupu DDS (bity 31:16) obsahuje signál sinus.



Obr. 2.25: Konfigurace DDS - výstupní frekvence

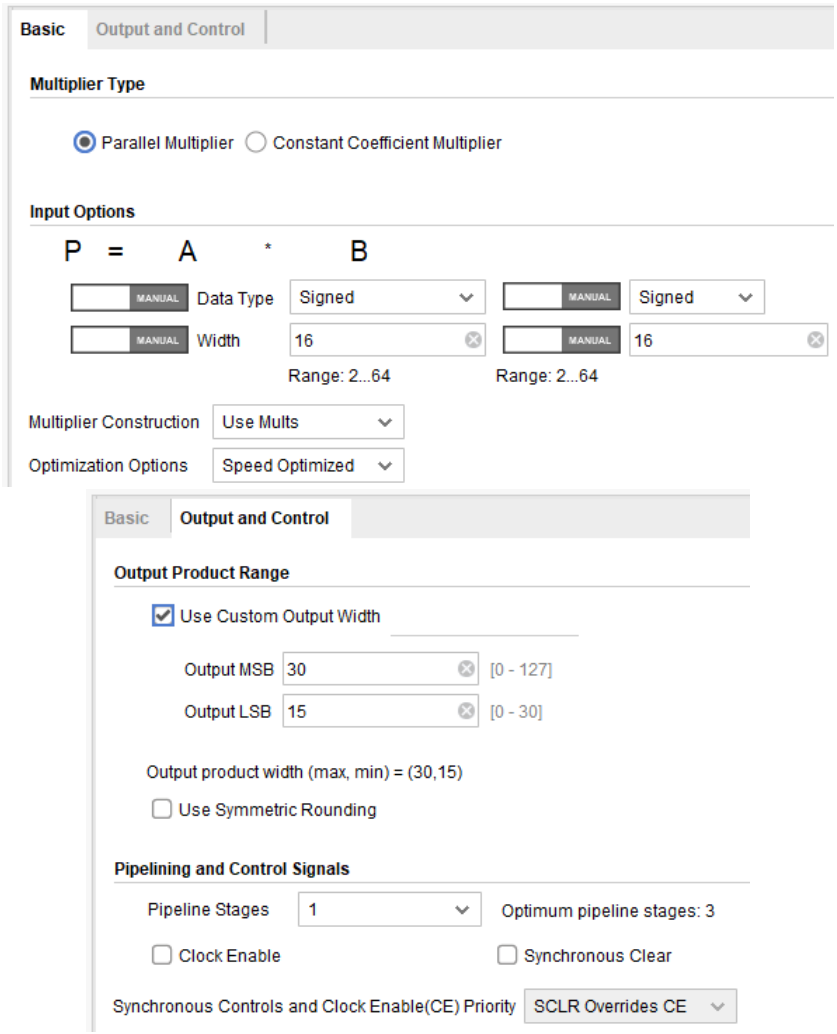
Vzhledem k tomu, že RP umožňuje přenos signálů v pásmu 0–50 MHz, byl zvolen výstupní kmitočet DDS 30 MHz (obr. 2.25). Výstupní signál z DDS je za pomoci bloku výběru bitů (obr. 2.26) rozdělen na dva datové toky. První datový tok vstupuje do násobičky N3 a druhý datový tok vstupuje do násobičky N4. Konfigurace bloků pro výběr bitů je vidět na obr. 2.27.



Obr. 2.26: Zapojení DDS a bloků pro výběr bitů

Obr. 2.27: Konfigurace bloku pro výběr bitů pro signál sinus a cosinus

Násobičky N3 a N4 (Multiplier)

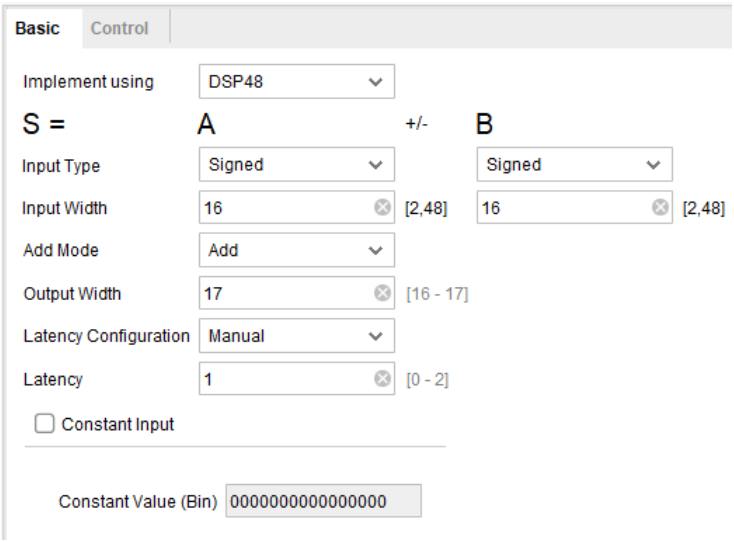


Obr. 2.28: Konfigurace Násobičky

Do násobičky vstupují 16 bitová znaménková data, podle čehož jsou také nastaveny operandy násobiček N3 a N4. Výstup je požadován také 16 bitový, proto je

výstupní rozsah bitů zvolen jak je vidět na obr. 2.28. Nejvyšší bit, který obsahuje znaménkové rozšíření po násobení není zahrnutý. Jestliže násobíme hodnotu ve formátu fix16_3 (výstup FIR interpolátoru) s hodnotou ve formátu fix16_15 (výstup DDS), po násobení pak dostaneme výsledek ve formátu fix32_18. Po vybrání bitů jak je na obr. 2.28 dostaneme hodnoty ve formátu fix16_3. Konfigurace násobiček byla prováděna dle [26].

Sčítačka (Adder/Subtracter)



Obr. 2.29: Konfigurace Sčítačky

Sčítačka [27] bude také pracovat s 16 bitovými znaménkovými operandy. Výstup pak bude 17 bitový v číselném formátu fix17_3. Po Sčítačce následuje blok pro výběr bitů, kterým se vybere pouze nejvyšších 16 bitů, které budou ve formátu fix16_2.

DAC (AXI4-Stream Red Pitaya DAC) [19]

- Šířka dat v bitech (AXIS TDATA WIDTH): 32 bitů
- Šířka dat pro DAC v bitech (DAC DATA WIDTH): 14 bitů

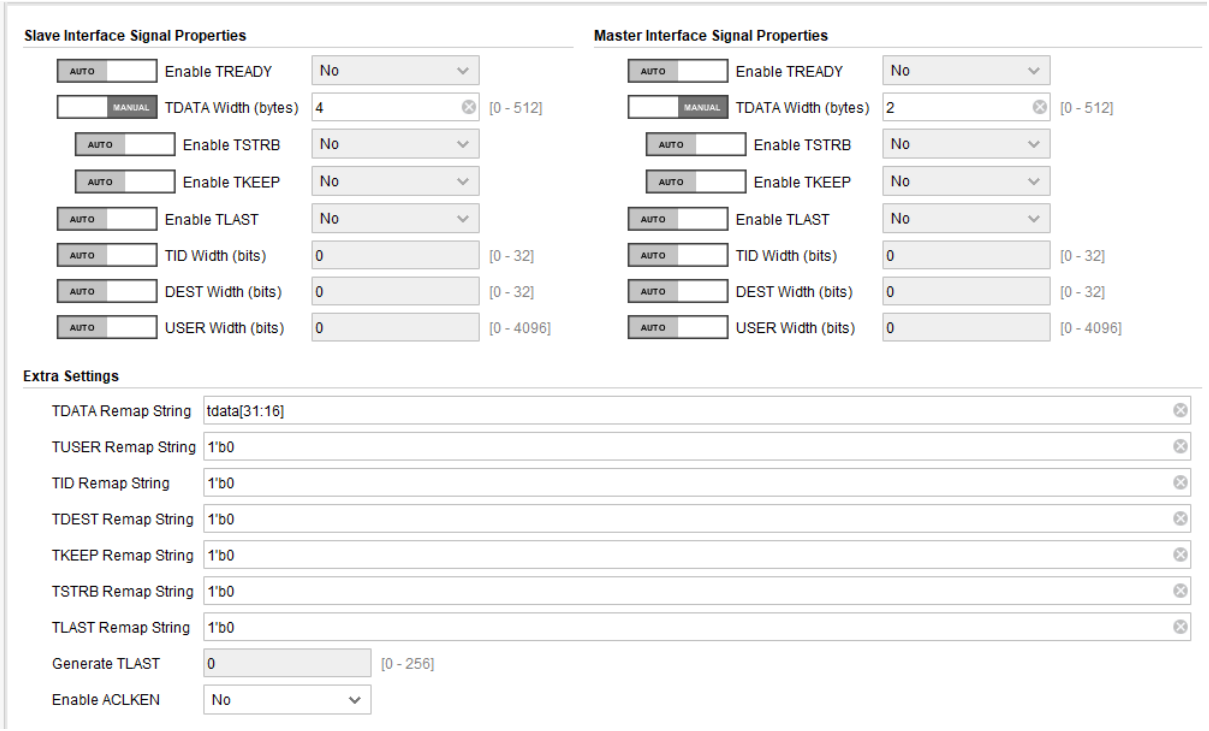
Samotnému Bloku DAC je předřazen blok dac_reg [29]. Tento blok zajišťuje přechod od bloků, které AXI4-Stream sběrnici nepodporují k těm, co ji podporují. Zároveň také dac_reg ořezává data, která přichází na jeho dva vstupy dac_ch0 a dac_ch1, na nejvyšších 14 bitů, které poté seshora doplní nulami. Z 16 bitových dat ve formátu fix16_2 se tak stanou 14 bitová data ve formátu fix14_0 a po doplnění nulami jsou ve formátu fix16_0. Tím jsou data připravena pro vstup do DAC.

ADC (AXI4-Stream Red Pitaya ADC) [19]

- Šířka dat v bitech (AXIS TDATA WIDTH): 32 bitů
- Šířka dat pro ADC v bitech (ADC DATA WIDTH): 14 bitů

ADC má na svém výstupu navzorkovaná data z obou kanálů. Spodních 16 bitů obsahuje vzorky prvního kanálu a horních 16 bitů obsahuje vzorky druhého kanálu.

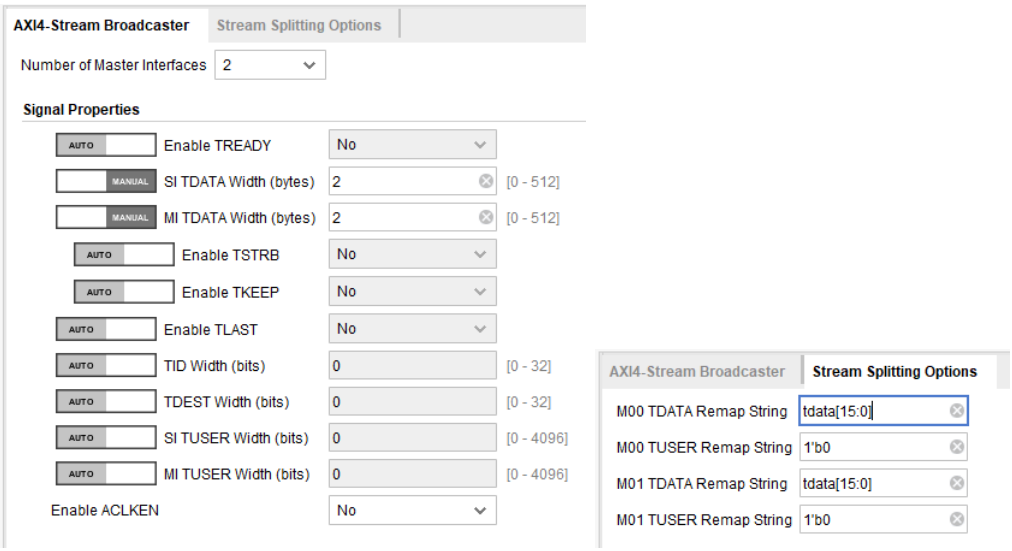
Převodník dat (AXI4-Stream Subset Converter)



Obr. 2.30: Konfigurace Převodníku podmnožin

Jak již bylo řečeno výše, ADC blok poskytuje data obou kanálů najednou. Jelikož vysíláme a přijímáme reálný signál namodulovaný na nosném kmitočtu 30 MHz na jednom kanálu DAC i ADC, je třeba použít převodník dat [22] pro výběr kanálu ADC, který je používán. Jak je vidět na obr. 2.30, vstupní šířka je nastavena na 32 bitů a výstupní šířka pak na 16 bitů. Mapovacím řetězcem zvolíme, že chceme vybrat data, která náleží kanálu 2 ADC.

Rozdělovač datového toku (AXI4-Stream Broadcaster)



Obr. 2.31: Konfigurace Převodníku podmnožin

Pro převod signálu do základního pásma je třeba signál navzorkovaný ADC rozdělit do dvou identických paralelních větví. Toho je docíleno použitím rozdělovače datového toku [22]. Vstupní i výstupní datová šířka je nastavena na 16 bitů. Na oba výstupy jsou pak mapovacím řetězcem přivedena totožná data.

Násobičky N1 a N2 (Multiplier)

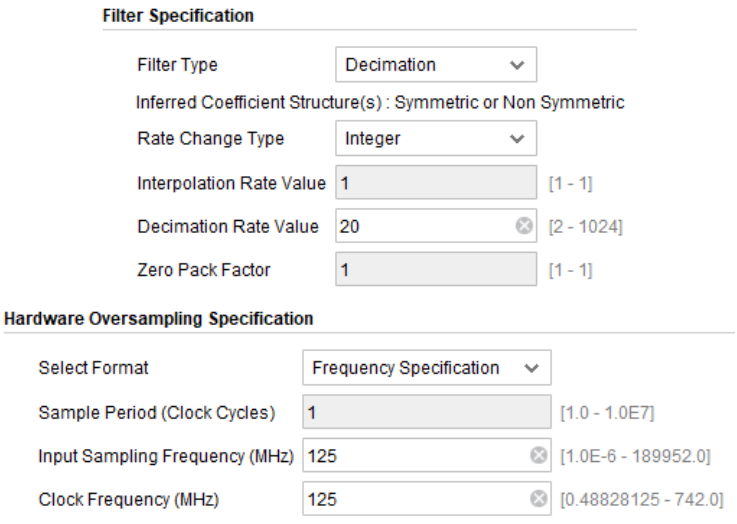
Konfigurace násobiček N1 a N2 je shodná s konfigurací násobiček N3 a N4 (obr. 2.28) podle [26].

FIR decimátory D1 a D2 (FIR Compiler)

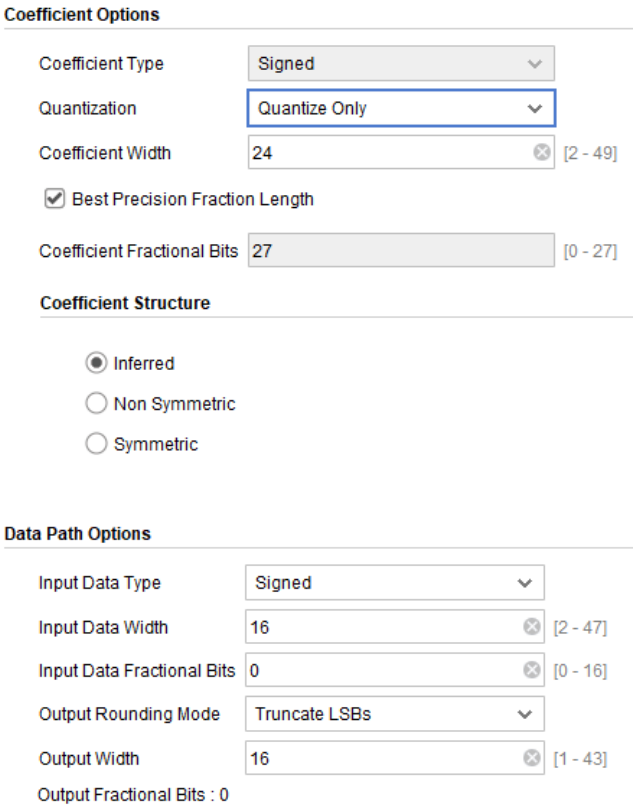
Pro decimátory D1 a D2 je použita stejná dolní propust jak pro interpolátory I1 a I2 (obr. 2.21). Konfigurace FIR kompilátoru podle [23] je na obr. 2.32 a obr. 2.33. Typ filtru je třeba nastavit na decimační s celočíselným faktorem 20. Decimační faktor lze spočítat podle vztahu

$$M = \frac{f_{vst}}{f_{výst}} = \frac{125 \cdot 10^6}{6.25 \cdot 10^6} = 20 \tag{2.8}$$

kde $f_{výst}$ je kmitočet po decimaci a f_{vst} je kmitočet před decimací.

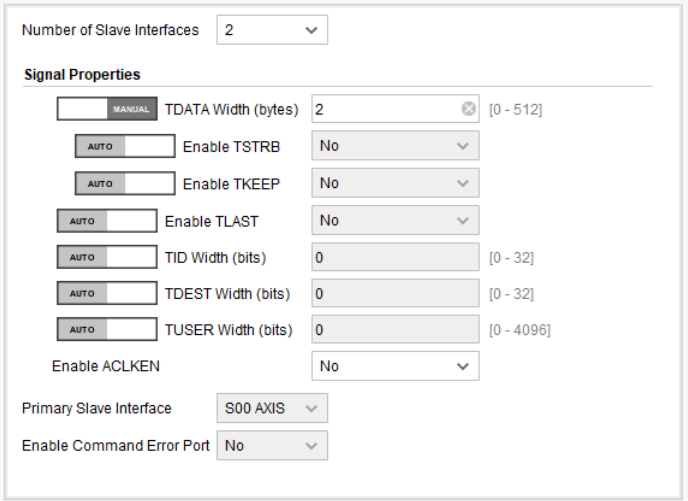


Obr. 2.32: Konfigurace FIR kompilátoru pro decimaci



Obr. 2.33: Konfigurace dat FIR kompilátoru pro decimaci

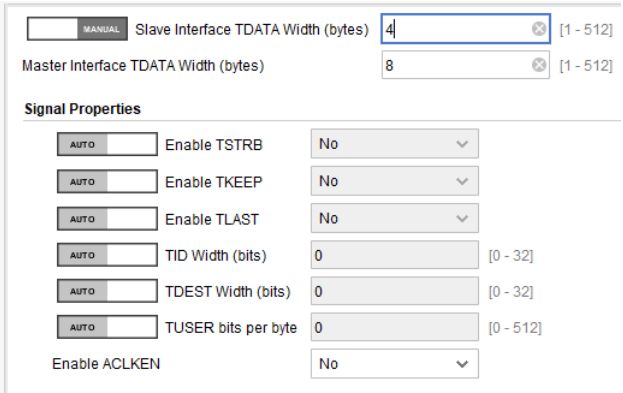
Slučovač datového toku (AXI4-Stream Combiner)



Obr. 2.34: Konfigurace Slučovače datového toku

Data převedená do základního pásma a decimovaná FIR decimátory D1 a D2 jsou dále sloučena do 32 bitového čísla, které se bude odesílat ke zpracování v Matlabu. Na první vstup slučovače datového toku [22] je třeba připojit datovou větev, která reprezentuje reálnou část a na druhý vstup datovou větev, která reprezentuje imaginární část dat. Bitová šířka každého ze vstupů je nastavena na 16 bitů. Výstupní bitová šířka je potom 32 bitů.

Převodník bitové šířky (AXI4-Stream Data Width Converter)



Obr. 2.35: Konfigurace Převodníku bitové šířky

Vstupní bitová šířka je nastavena na 32 bitů, výstupní bitová šířka na 64 bitů podle [22].

Zapisovač (AXI4-Stream RAM Writer) [19]

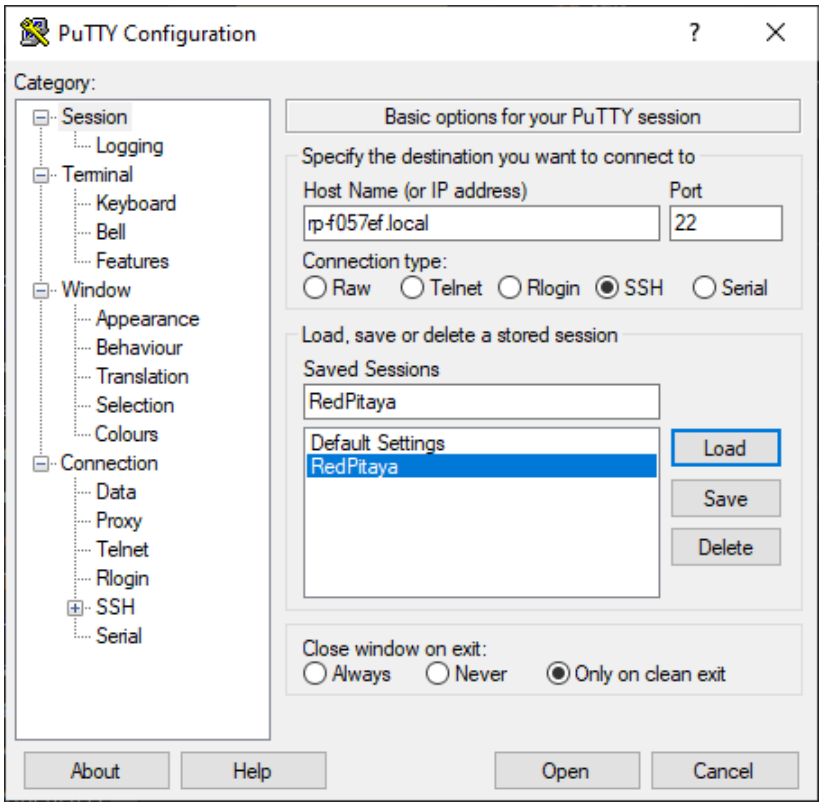
- Šířka adresy v bitech (AXI ADDR WIDTH): 16 bitů
- Šířka dat v bitech (AXI DATA WIDTH): 32 bitů

2.4 Činnost generátoru a výsledky

Tato kapitola se zabývá připojením k vývojovému kitu STEMLab RedPitaya, nahráním bitstreamu designu navrženého v programu Vivado, spuštěním C serveru. Jsou zde také zmíněny parametry generovaného signálu a zobrazeny obrázky generovaného a přijatého signálu.

2.4.1 RedPitaya - připojení a nahrání bitstreamu

Vývojový kit je třeba připojit Ethernetovým kabelem k počítači. Po připojení Ethernetového kabelu a připojení k napájení je možné se do kitu připojit například pomocí programu Putty. IP adresa je ve tvaru rp-XXXXXX.local, kde za XXXXXX je třeba doplnit posledních 6 znaků z fyzické adresy kitu [18]. V případě mnou použitého konkrétního kitu se jedná o IP adresu rp-f057ef.local a číslo portu je 22 (obr.2.36). Po vyzvání je třeba se přihlásit přihlašovacím jménem: root a heslem: root.



Obr. 2.36: Konfigurace Putty pro připojení k Red Pitaya

Vygenerovaný bitstream z programu Vivado a C server je třeba překopírovat do kitu. K tomu může sloužit program WinSCP, který má způsob připojení ke kitu Red

Pitaya obdobné jako Putty. Soubory je pak třeba překopírovat do adresáře /root, ve kterém se budou zároveň zadávat veškeré příkazy. C server je možné přeložit přímo v Putty. Obecný způsob jak zkompilovat C soubor je pomocí příkazu:

`gcc <jmeno_souboru>.c -o <jmeno_souboru> [parametry_překladu]`.

V případě souboru `adc_dac_server.c` je pak příkaz upraven na:

`gcc adc_dac_server.c -o adc_dac_server -lm`.

Bitstream nakopírovaný do RP lze spustit příkazem:

`cat adc_mtlb_dac3_7.bit > /dev/xdev`.

Přeložený server je třeba spustit příkazem:

`./adc_dac_server`.

Server je možné kdykoliv vypnout kombinací kláves Ctrl+C. Nyní je Red Pitaya připravena k příjmu dat z Matlabu.

2.4.2 Generování a příjem dat

Testovací OFDM signál má následující parametry

- Modulace: 16QAM
- Počet datových subnosných: 324
- Velikost FFT: 512
- Vzdálenost mezi nosnosnými Δf: 15 kHz
- Vzorkovací kmitočet f_{vz}: 6.25 MHz
- DMRS konfigurace: Na všech subnosných a současně na prvním symbolu

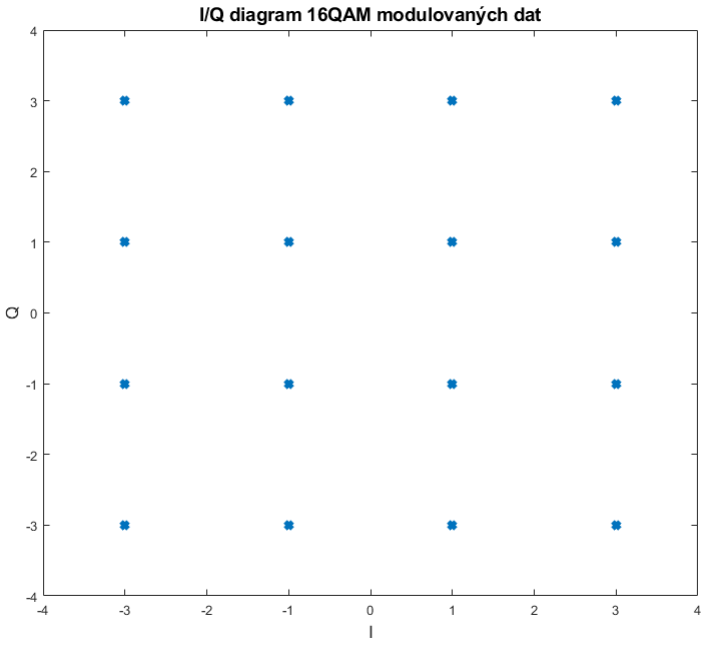
Vzorkovací frekvence napsaná v parametrech je jiná, než by měla v závislosti na velikosti FFT a vzdálenosti nosných ve skutečnosti být. To je způsobeno tím, že se FIR filtry v kapitole 2.3.3 nepodařilo nakonfigurovat na neceločíselný interpolační a decimační faktor. Na funkčnost generátoru to však nemá významný vliv. Mezi vzorkovací frekvencí f_{vz} a vzdáleností mezi nosnými Δf platí následující vztah:

$$f_{vz} = NFFT \cdot \Delta f \tag{2.9}$$

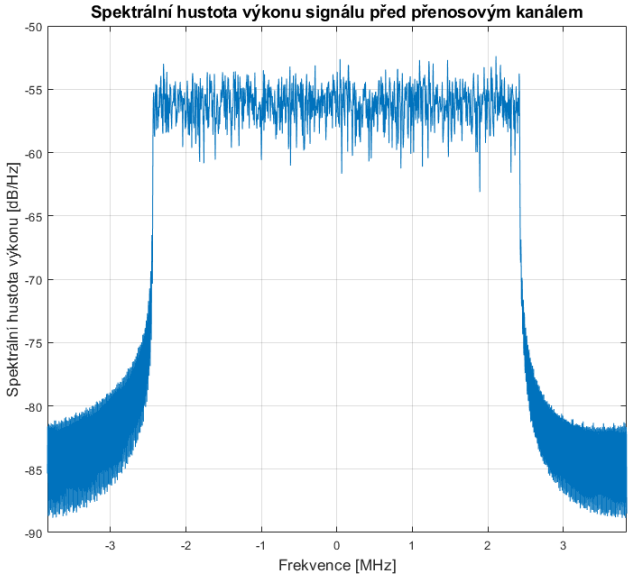
kde f_{vz} je vzorkovací kmitočet, $NFFT$ je velikost FFT a Δf je vzdálenost mezi nosnými,

Po spuštění skriptu `ofdm_mod2_txrx3_6.m` proběhne vygenerování signálu podle výše uvedených parametrů. Konstelační diagram a spektrální hustota výkonu generovaného (a tedy vysílaného) signálu v základním pásmu je vidět na obr. 2.37 a na obr. 2.38. Vygenerovaná data vstupují v podobě vektoru do funkce `f_rp_tran_acq.m`, která zajistí jejich odeslání pomocí TCP na server. S pomocí serveru budou data dále poslána do RP.

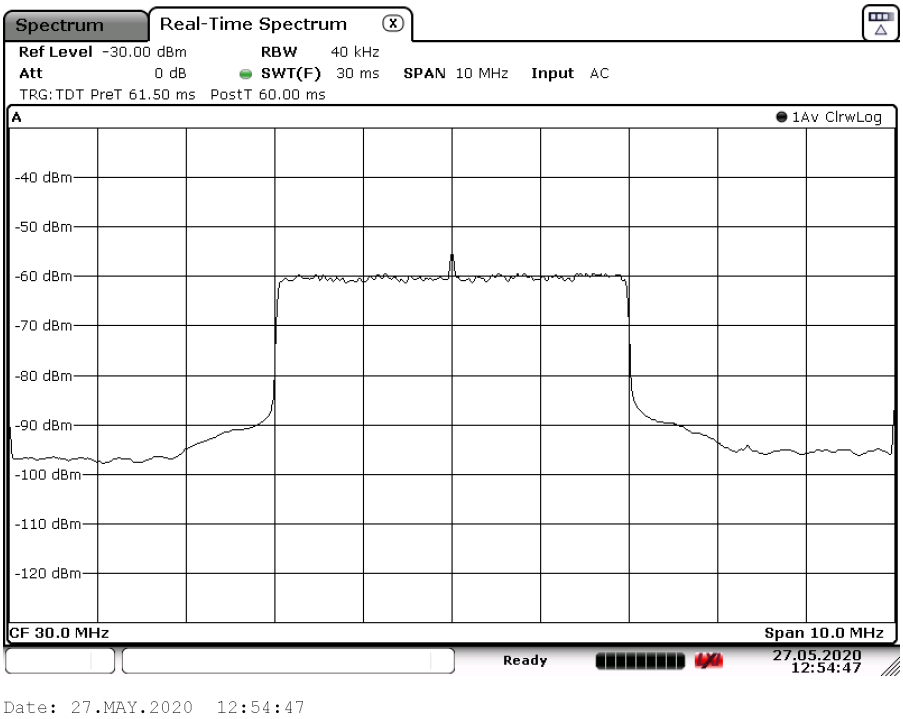
Na obr. 2.39 je vidět odhad spektra na výstupu DAC.



Obr. 2.37: I/Q diagram 16QAM modulovaných dat

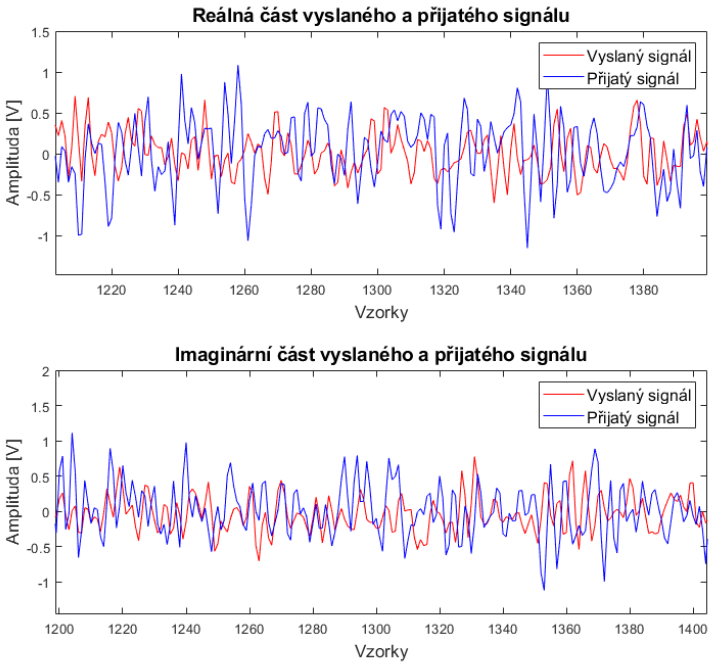


Obr. 2.38: Spektrální hustota výkonu vysílaných dat

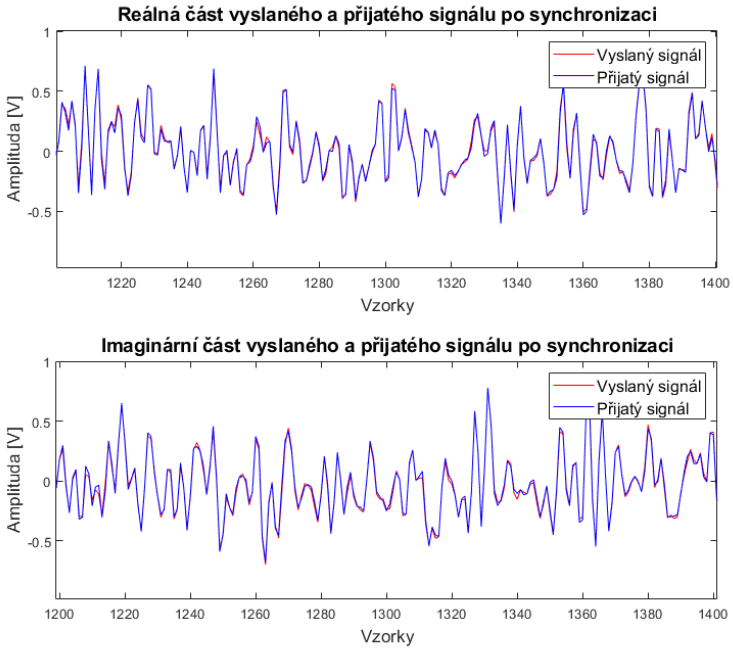


Obr. 2.39: Odhad spektra dat na výstupu DAC změřený real–time spektrálním analyzátorem FSVR

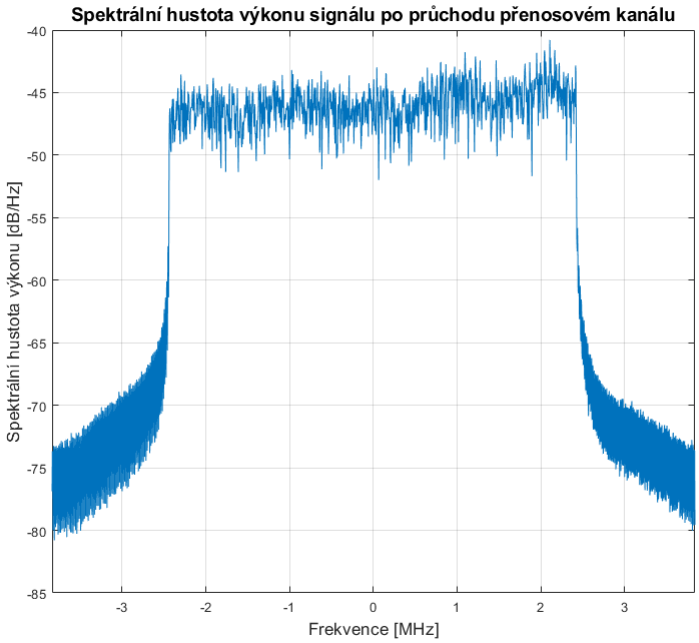
Data, která prošla komunikačním řetězcem v RP jsou přes server přijata opět funkcí `f_rp_tran_acq.m`. V důsledku šíření signálu celým komunikačním řetězcem, včetně kabelu smyčky, vzniká zpoždění mezi vyslaným a přijatým signálem, které není nezanedbatelné. Na obr. 2.40 je vidět úsek vyslaného signálu a signálu přijatého se zpožděním. Horní graf zobrazuje reálnou část signálu, spodní graf zobrazuje imaginární část signálu. Signál po časové synchronizaci je vidět na obr. 2.41.



Obr. 2.40: Reálná a imaginární složka dat před časovou synchronizací



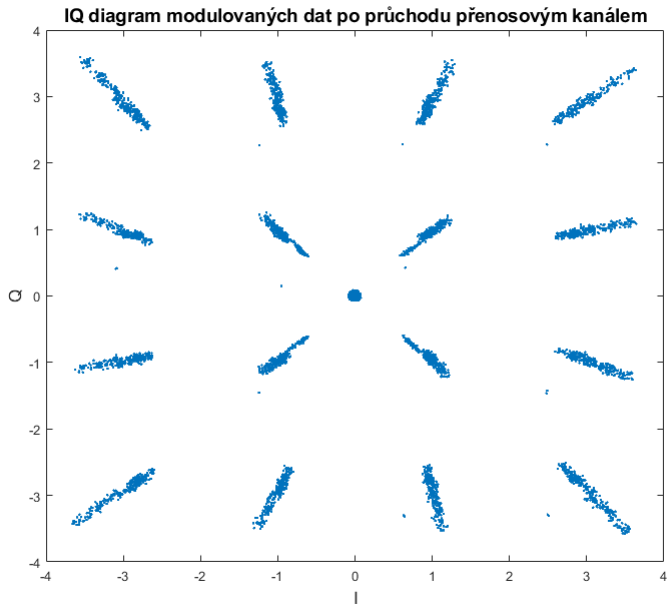
Obr. 2.41: Reálná a imaginární složka dat po časové synchronizaci



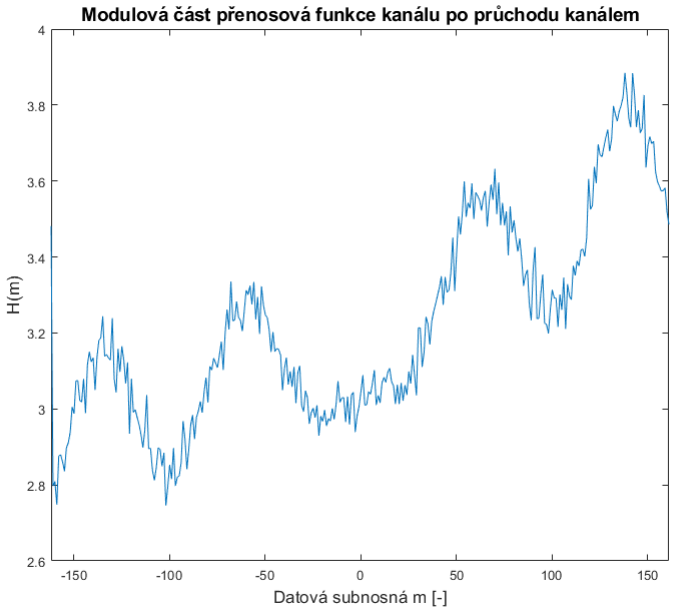
Obr. 2.42: Spektrální hustota výkonu přijatého signálu

Na obr. 2.42 je vidět spektrální hustota výkonu a na obr. 2.43 konstelační diagram přijatého zasynchronizovaného signálu. Je zřejmé, že signál není amplitudově konstnatní (zvlnění frekvenčního spektra a viditelné protažení symbolů v konstelačním diagramu). To z velké části způsobuje FIR interpolátor a decimátor, respektive jejich dolní propust, která nemá v propustném pásmu nulové zvlnění.

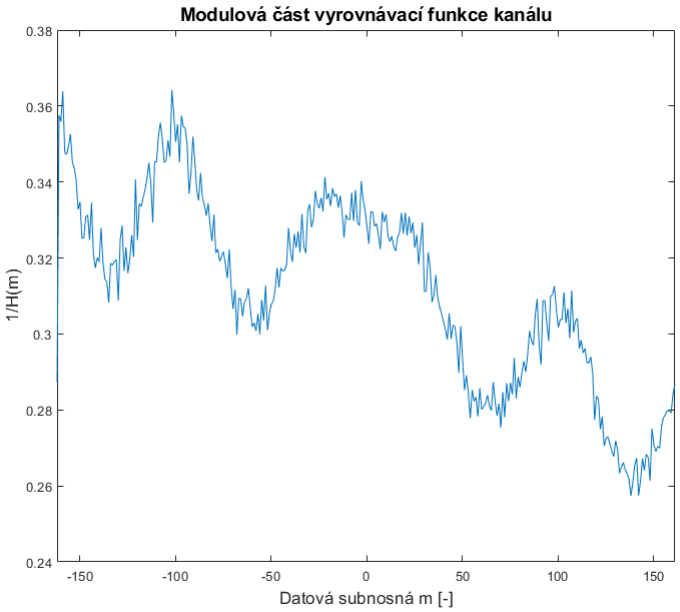
Vypočítáním přenosové funkce kanálu (obr.2.44) z DMRS jsme schopni určit vyrovnávací funkci kanálu (obr.2.45) s jejíž pomocí můžeme vyrovnat zvlnění amplitudy signálu. Byl použit Zero-Forcing vyrovnávač (viz. kapitola 1.1.9). Konstelační diagram signálu po úpravě vyrovnávací funkcí je vidět na obr. 2.46.



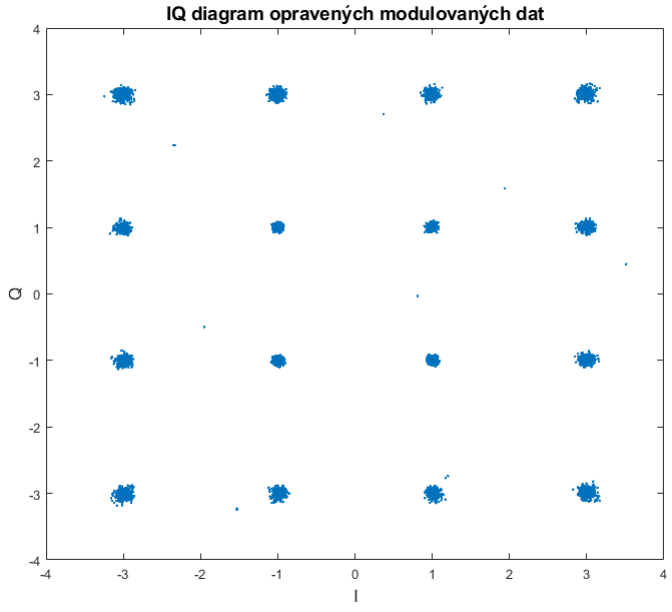
Obr. 2.43: I/Q diagram 16QAM modulovaných dat po průchodu přenosovým kanálem bez úpravy vyrovnávačem



Obr. 2.44: Modulová část přenosové funkce kanálu



Obr. 2.45: Modulová část vyrovnávací funkce kanálu



Obr. 2.46: I/Q diagram opravených 16QAM modulovaných dat

3 ZÁVĚR

V první části práce jsem prostudoval fyzickou vrstvu připravovaného 5G standardu (Rel.15). Pro implementaci generátoru byl v rámci semestrálního projektu v souladu s jeho zadáním zvolen vývojový kit od firmy Xilinx s označením Zynq UltraScale+ RFSoC ZCU111. Kit by sloužil zároveň pro vysílání i příjem pomocí zpětné smyčky (loopbacku). V souvislosti s vývojovým kitem jsem prostudoval jeho technickou dokumentaci se zaměřením na RF Data Converter, který obsahuje důležité části pro pokračování práce.

V programu Matlab jsem úspěšně nasimuloval základní strukturu OFDM modulátoru a demodulátoru, jejíž parametry se mění v závislosti na použité numerologii (například vzdálenosti mezi nosnými a počtu nosných). Pro simulaci bylo nutno vytvořit vlastní funkci sloužící pro vložení a odstranění různě dlouhých cyklických prefixů. Vyrovnávač kanálu typu Zero Forcing, který byl přidán do simulace je schopný vyrovnat přenosovou charakteristiku kanálu za pomoci zjednodušených DMRS.

Navrhl jsem koncepci výsledného generátoru 5G signálů včetně parametrů, které je třeba nastavit převážně v RF Data Converteru vývojového kitu. První koncepce spočívá ve vytvoření fyzické vrstvy v simulačním programu a úpravě pro vysílání v RFDC vývojového kitu. Druhá koncepce navrhuje vytvoření fyzické vrstvy včetně úpravy signálu pro vysílání ve vývojovém kitu. Ověření funkce generátoru by v souvislosti s tím probíhalo buď v simulačním programu nebo ve vývojovém kitu.

Jelikož jsem počítal s implementací generátoru do kitu ZCU111, práce obsahuje jeho popis a konfiguraci, ačkoliv tento nakonec z důvodu pozdního dodání nebyl pro implementaci generátoru použit. Místo něj byl, taktéž v souladu se zadáním diplomové práce, zvolen vývojový kit STEMLab RedPitaya.

Do vývojového kitu STEMLab RedPitaya jsem úspěšně implementoval zjednodušený generátor 5G OFDM signálu. Implementovaný generátor je schopen pomocí TCP spojení přijmout data v základní pásmu, provést jejich interpolaci a převést je na nosný kmitočet. Vzhledem k parametrům vývojového kitu (především maximálnímu kmitočtu 50 MHz) a k šířce pásma generovaných signálů, byl zvolen nosný kmitočet 30 MHz. Pro převod signálu na frekvence definované 5G standardem by bylo třeba využít dodatečných externích obvodů (směšovačů). Zároveň je generátor schopen vygenerovaný reálný signál decimovat, převést do základního pásma a odeslat pomocí TCP spojení k demodulaci a vyhodnocení. Vygenerovaný signál je pro jednoduchost vracen zpět do vývojového kitu pomocí zpětné smyčky (propojení DAC a ADC za pomoci kabelu).

V prostředí Matlab byl vytvořen softwarový demodulátor, který je schopen přijmout data pomocí TCP spojení. Přijatá data je schopen následně časově synchronizovat, opravit pomocí Zero Forcing vyrovnávače, demodulovat a určit bitovou chy-

bovost mezi daty vyslanými a daty přijatými. Softwarový demodulátor v prostředí Matlab zároveň slouží pro generování dat v základním pásmu, která jsou pomocí TCP spojení odesílána do kitu RedPitaya pro další zpracování. Komunikaci mezi Matlabem a implementovaným designem v PL čipu XC7Z010 na vývojovém kitu RedPitaya zprostředkovává server v jazyku C.

Bitová chybovost přijímaného signálu dosahuje chybovosti $8,3 \cdot 10^{-4} (83 \cdot 10^{-3}$ %).

Z časových důvodů byly použity FIR interpolátory a decimátory pouze s celočíselným faktorem. Z toho důvodu není dosaženo přesné vzorkovací frekvence (7,68 MHz pro danou konfiguraci) dle 5G standardu. Požadované vzorkovací frekvence by mohlo být dosaženo, pokud bychom interpolační/decimační faktor FIR filtrů nastavili jako neceločíselný. Bylo by také možné použít zapojení dvou FIR interpolátorů nebo decimátorů do kaskády.

LITERATURA

[1] *QuickReference - 5G/NR* [online]. ShareTechnote. [cit. 24. 9. 2019] Dostupné z URL: <https://www.sharetechnote.com/html/5G/Handbook_5G_Index.html>.

[2] *5G; NR; Physical channels and modulation* [online]. ETSI 2018 [cit. 24. 9. 2019] Dostupné z URL: <https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/15.02.00_60/ts_138211v150200p.pdf>.

[3] *Understanding the 5G NR Physical Layer* [online]. Keysight Technologies 2017. [cit. 24. 9. 2019] Dostupné z URL: <https://www.keysight.com/upload/cmc_upload/All/Understanding_the_5G_NR_Physical_Layer.pdf>.

[4] *5G New Radio - Fundamentals, procedures, testing aspects* [online]. Rohde & Schwarz 2019. [cit. 24. 9. 2019] Dostupné z URL: <https://www.rohde-schwarz.com/us/solutions/test-and-measurement/wireless-communication/wireless-5g-and-cellular/5g-ebook/5g-nr-ebook_250786.html>.

[5] *Mathematical description of OFDM* [online]. Dusan Matic, 1999. [cit. 24. 9. 2019] Dostupné z URL: <http://www.wirelesscommunication.nl/reference/chaptr05/ofdm/ofdmmath.htm>.

[6] *5G NR Bandwidth Part (BWP)* [online]. telecomHall Forum. [cit. 24. 9. 2019] Dostupné z URL: <http://www.telecomhall.net/t/5g-nr-bandwidth-part-bwp/4570>.

[7] *ZCU111 Evaluation Board, User Guide* [online]. Xilinx 2018. [cit. 8. 10. 2019] Dostupné z URL: <https://www.xilinx.com/support/documentation/boards_and_kits/zcu111/ug1271-zcu111-eval-bd.pdf>.

[8] *Zynq UltraScale+ RFSoC - Product Tables and Product Selection Guide* [online]. Xilinx 2018. [cit. 8. 10. 2019] Dostupné z URL: <https://www.xilinx.com/support/documentation/selection-guides/zynq-usp-rfsoc-product-selection-guide.pdf>.

[9] *Zynq UltraScale+ RFSoC RF Data Converter Evaluation Tool (ZCU111), User Guide* [online]. Xilinx 2018. [cit. 8. 10. 2019] Dostupné z URL: <https://www.xilinx.com/support/documentation/boards_and_kits/zcu111/2018_3/ug1287-zcu111-rfsoc-eval-tool.pdf>.

[10] *Zynq UltraScale+ RFSoC RF Data Converter 2.1, LogiCORE IP Product Guide* [online]. Xilinx 2018. [cit. 28. 10. 2019] Dostupné z URL: <https://www.xilinx.com/support/documentation/ip_documentation/usp_rf_data_converter/v2_1/pg269-rf-data-converter.pdf>.

[11] *Understanding the 5G NR Standard* [online]. 1994-2019 The MathWorks, Inc. [cit. 16. 10. 2019] Dostupné z URL: <https://www.mathworks.com/videos/series/understanding-the-5g-nr-standard.html>.

[12] MARŠÁLEK Roman. *Teorie rádiové komunikace*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2012. ISBN 978-80-214-4503-1. [cit. 3. 12. 2019].

[13] *Zynq-7000 SoC Data Sheet: Overview* [online]. 2012–2018 Xilinx, Inc. [cit. 27. 5. 2020] Dostupné z URL: <https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf>.

[14] *Zynq-7000 SoC Technical Reference Manual* [online]. 2012-2017 Xilinx, Inc. [cit. 27. 5. 2020] Dostupné z URL: <https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf>.

[15] *RED PITAYA STEMLAB BOARD 125-14* [online]. 2020 StemLabs [cit. 27. 5. 2020] Dostupné z URL: <https://www.redpitaya.com/194/Red%20Pitaya%20STEMlab%20board%20125-14>.

[16] *Red Pitaya Electrical schematics* [online]. 2014, Red Pitaya d.o.o. [cit. 27. 5. 2020] Dostupné z URL: <https://redpitaya.readthedocs.io/en/latest/developerGuide/125-14/shem.html>.

[17] *Red Pitaya OS* [online]. 2017, Red Pitaya [cit. 27. 5. 2020] Dostupné z URL: <https://redpitaya.readthedocs.io/en/latest/developerGuide/os/debian.html>.

[18] *Connect to your Red Pitaya* [online]. 2017, Red Pitaya [cit. 27. 5. 2020] Dostupné z URL:
<https://redpitaya.readthedocs.io/en/latest/quickStart/connect/connect.html>.

[19] *Red Pitaya Notes* [online]. Demin Pavel [cit. 27. 5. 2020] Dostupné z URL:
<https://github.com/pavel-demin/red-pitaya-notes>.

[20] *Red Pitaya Notes - SDR transciever* [online]. Demin Pavel [cit. 27. 5. 2020] Dostupné z URL:
<https://github.com/pavel-demin/red-pitaya-notes/tree/master/projects/sdr_transceiver>.

[21] *FIFO Generator v13.2* [online]. 2012-2017 Xilinx, Inc. [cit. 27. 5. 2020] Dostupné z URL:
<https://www.xilinx.com/support/documentation/ip_documentation/fifo_generator/v13_2/pg057-fifo-generator.pdf>.

[22] *AXI4-Stream Infrastructure IP Suite v3.0* [online]. 2012-2018 Xilinx, Inc. [cit. 27. 5. 2020] Dostupné z URL:
<https://www.xilinx.com/support/documentation/ip_documentation/axis_infrastructure_ip_suite/v1_1/pg085-axi4stream-infrastructure.pdf>.

[23] *CIC Compiler v4.0* [online]. 2012-2016 Xilinx, Inc. [cit. 27. 5. 2020] Dostupné z URL:
<https://www.xilinx.com/support/documentation/ip_documentation/cic_compiler/v4_0/pg140-cic-compiler.pdf>.

[24] *Use Filter Designer with DSP System Toolbox Software* [online]. 1994-2020 The MathWorks, Inc. [cit. 27. 5. 2020] Dostupné z URL:
<https://www.mathworks.com/help/dsp/ug/use-fdatool-with-dsp-system-toolbox-s
html>.

[25] *DDS Compiler v6.0* [online]. 2013–2017 Xilinx, Inc. [cit. 27. 5. 2020] Dostupné z URL:
<https://www.xilinx.com/support/documentation/ip_documentation/dds_compiler/v6_0/pg141-dds-compiler.pdf>.

[26] *Multiplier v12.0* [online]. 2013–2015 Xilinx, Inc. [cit. 27. 5. 2020] Dostupné z URL:
<https://www.xilinx.com/support/documentation/ip_documentation/mult_gen/v12_0/pg108-mult-gen.pdf>.

[27] *Adder/Subtracter v12.0* [online]. 2012–2015 Xilinx, Inc. [cit. 27. 5. 2020] Dostupné z URL:
<https://www.xilinx.com/support/documentation/ip_documentation/addsub/v12_0/pg120-c-addsub.pdf>.

[28] J. Kral, T. Gotthans and M. Harvanek,
Analytical method of fractional sample period synchronisation for digital predistortion systems [online]. 2017 27th International Conference Radioelektronika (RADIOELEKTRONIKA), Brno, 2017, pp. 1-5, doi: 10.1109/RADIOELEK.2017.7937603 [cit. 27. 5. 2020] Dostupné z URL:
<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7937603>.

[29] R. Maršálek, M. Waldecker,
Implementace softwarových komunikačních systémů - BPSK [online]. [cit. 27. 5. 2020] Dostupné z URL:
<https://moodle.vutbr.cz/mod/folder/view.php?id=36628>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ADC	Analog to Digital Converter
APU	Aplication Processor Unit
AWGN	Additive White Gaussian Noise
AXI	Advanced eXtensible Interface
BB	Base Band
BP	Band Pass
BPSK	Binary Phase Shift Keying
BRAM	Block RAM
BWP	BandWidth Part
CIC	Cascaded Integrator-Comb
CLB	Configurable Logic Block
CP	Cyclic Prefix
CRB	Common Resource Block
CSI	Channel State Information
CSIRS	Channel State Information Reference Signal
DAC	Digital to Analog Converter
DDC	Digital Down Converter
DDR	Double-Data Rate
DDS	Direct Digital Synthesizers
DFE	Decision Feedback Equalizer
DL	Downlink
DMA	Direct Memory Access
DMRS	DeModulation Reference Signal
DSP	Digital Signal Processing
DUC	Digital Up Converter
EMIO	Extended MIO
FIFO	First-In First-Out
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
gNB	Next generation NodeB
GP	General Purpose port
HP	High Performance port
IFFT	Inverse Fast Fourier Transform
IOP	I/O Peripherals
ISI	InterSymbol Interference
I2C	Inter-Integrated Circuit
LDPC	Low Density Parity Check

LP	Low Pass
LTE	Long Term Evolution
LUT	Look-Up Table
MEX	Matlab Executable
MIMO	Multiple Input Multiple Output
MIO	Multiplexed I/O
MMCM	Mixed-Mode Clock Manager
MMSE	Minimum Mean Square Error
NCO	Numerical Controlled Oscillator
OFDM	Orthogonal Frequency Division Multiplexing
PDSCH	Physical Downlink Shared Channel
PFD	Phase Frequency Detector
PL	Programmable Logic
PLL	Phase Locked Loop
PRB	Physical Resource Block
PS	Processing System
PTRS	Phase Tracking Reference Signal
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory
RB	Resource Block
RE	Resource Element
RFDC	Radio Frequency Data Converter
RG	Resource Grid
RP	Red Pitaya
SFDR	Spurious Free Dynamic Range
SG DMA	Scatter Gather DMA
SISO	Single Input Single Output
SNR	Signal-to-Noise Ratio
SPI	Serial Peripheral Interface
SRS	Sounding Reference Singal
TCP	Transmission Control Protocol
UE	User Equipment
UL	Uplink
VCO	Voltage Control Oscillator

SEZNAM PŘÍLOH

A Tabulky ke konfiguraci RFDC **93**

B Blokové schéma implementace **95**

A TABULKY KE KONFIGURACI RFDC

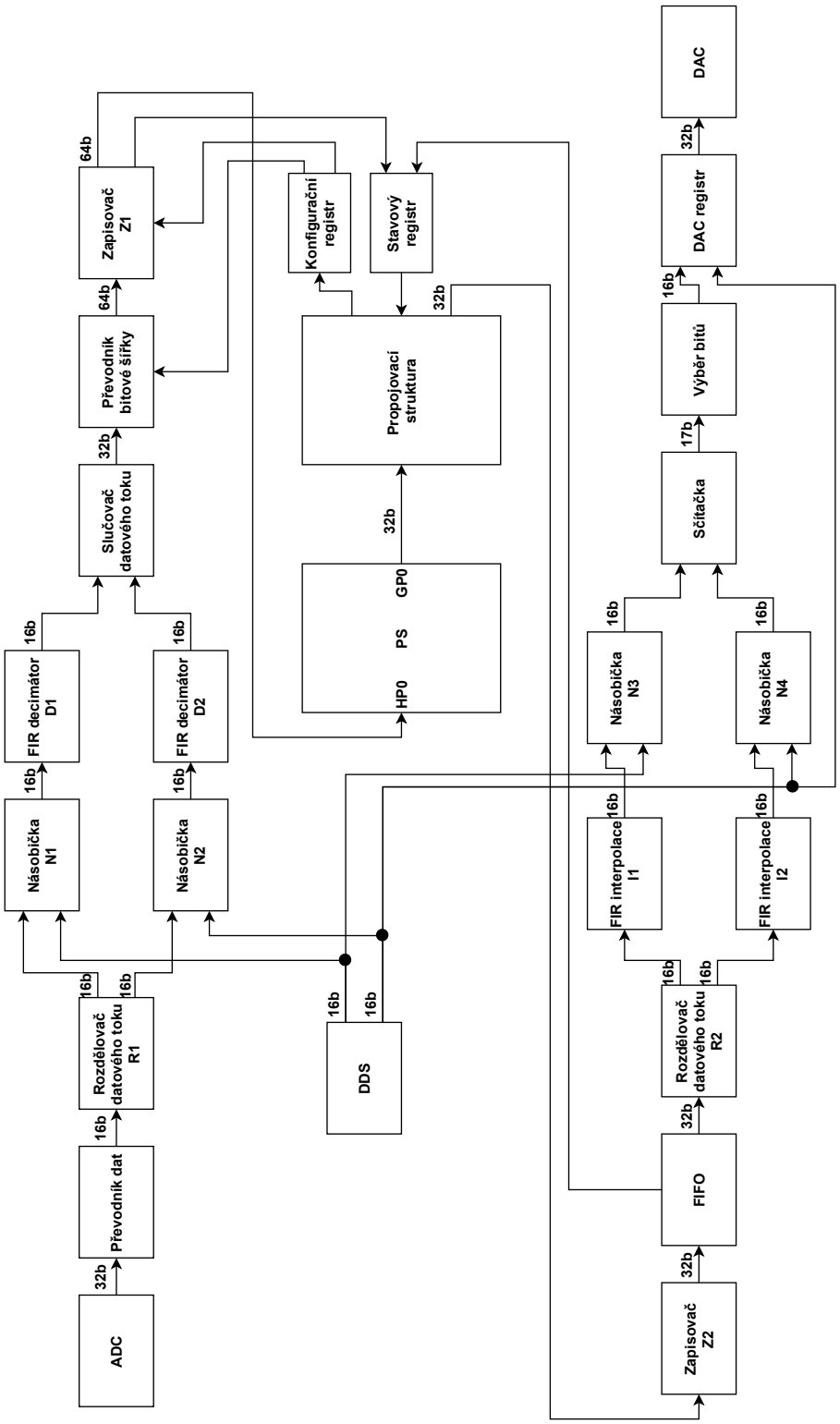
Název parametru	Varianta		
	První	Druhá	Třetí
Link Coupling	AC	AC	-
Converter Band Mode	Single	Single	-
Enable ADC	ADC 0, ADC 1	ADC 0, ADC1	NO
Invert Q Output	NO	NO	-
Dither	YES	YES	-
Bypass Background Calibration	NO	NO	-
Digital Output Data	I/Q	I/Q	-
Decimation Mode	x1	x1	-
Samples per AXI4-Stream Word	8	8	8
Mixer Type	Fine	Fine	Fine
Mixer Mode	Real -> I/Q	Real -> I/Q	Real -> I/Q
Coarse Mixer Frequency	-	-	-
Fine Mixer Frequency (GHz)	-0,4	-0,4	-2,1
Fine Mixer Phase	0	0	0
Nyquist Zone	Zone 2	Zone 2	-
Calibration Mode	Mode 2	Mode 2	-
Sampling Rate (GSPS)	4	4	-
PLL	NO	NO	NO
Reference Clock (MHz)	4	4	4
Clock Out (MHz)	-	-	-

Tab. A.1: Zvolená konfigurace pro ADC

Název parametru	Varianta		
	První	Druhá	Třetí
Converter Band Mode	Single	Single	-
Enable DAC	DAC 0	DAC 0	NO
Invert Q Output	NO	NO	-
Inverse Sinc Filter	YES	YES	-
Analog Output Data	Real	Real	-
Interpolation Mode	x1	x1	-
Samples per AXI4-Stream Word	8	8	8
Mixer Type	Fine	Fine	Fine
Mixer Mode	I/Q -> Real	I/Q -> Real	I/Q -> Real
Coarse Mixer Frequency	-	-	-
Fine Mixer Frequency (GHz)	2,9	0,4	2,1
Fine Mixer Phase	0	0	0
Nyquist Zone	Zone 2	Zone 2	-
Decoder Mode	SNR Optimized	SNR Optimized	-
Sampling Rate (GSPS)	6,5	6,5	-
PLL	NO	NO	-
Reference Clock (MHz)	6,5	6,5	-
Clock Out (MHz)	-	-	-

Tab. A.2: Zvolená konfigurace pro DAC

B BLOKOVÉ SCHÉMA IMPLEMENTACE



Obr. B.1: Blokové schéma implementace v programu Vivado